Université Paris 7 – Denis Diderot
U.F.R. d'Informatique

Habilitation à diriger des recherches

présentée par

Sylvain LOMBARDY

Contribution aux problèmes de réalisation des langages et séries rationnels

Soutenue publiquement le 6 décembre 2005

devant le jury composé de

Marie-Pierre BÉAL
Christian CHOFFRUT, *Rapporteur*
Erich GRÄDEL
Jean MAIRESSE
Antonio RESTIVO
Jacques SAKAROVITCH, *Directeur*
Igor WALUKIEWICZ, *Rapporteur*

Remerciements

Je tiens à remercier en premier lieu Jacques Sakarovitch qui m'a soutenu depuis mes débuts dans le métier de chercheur et plus particulièrement durant la préparation de cette habilitation.

Je remercie Christian Choffrut, Juraj Hromkovitch, Christophe Reutenauer et Igor Walukiewicz d'avoir acceptés d'être rapporteurs sur ce mémoire d'habilitation.

Je remercie Marie-Pierre Béal, Christian Choffrut, Erich Grädel, Jean Mairesse, Antonio Restivo, Jacques Sakarovitch et Igor Walukiewicz d'avoir acceptés d'être membres du jury de cette soutenance.

Je remercie Christophe, Ines, Jacques, Jean, Thomas, Louis-Noël, Marie-Pierre, Raphaël, Sarah et Yann pour toutes les discussions échangées pendant que nous écrivions des articles.

Je remercie tout le Vaucanson groupe pour sa disponibilité, les membres du LIAFA et de l'UFR d'informatique de Paris 7 pour l'ambiance qui y règne.

Je remercie mes amis pour leur fidélité et ma famille qui me supporte dans tous les sens du terme.

Je remercie Cécile pour la vie au quotidien à ses côtés.

Table des matières

Introduction

Ce mémoire présente les travaux de recherche que j'ai effectués jusqu'à présent. Il a pour ambition de montrer les principaux résultats ainsi que les algorithmes correspondants. Il se veut suffisant à lui-même, mais on devra se référer aux articles indiqués pour trouver toutes les preuves et les détails techniques.

Hormis les travaux portant sur les automates max-plus, une majeure partie de cette recherche a été effectuée en collaboration avec Jacques Sakarovitch.

Le dénominateur commun de ces travaux est la recherche de structure au sein des objets étudiés. De nombreux problèmes de la théorie des automates sont en effet résolus en se projetant dans des structures finies (monoïde syntaxique, ensemble de matrices dans un semi-anneau fini, *etc.*) dans lesquels il suffit de faire une recherche exhaustive pour vérifier telle ou telle propriété : limitation des séries, hauteur d'étoile, hauteur d'étoile généralisée nulle, *etc.* Toutefois, sans dénigrer la beauté et la puissance de ces preuves, on peut leur reprocher de s'abstraire complètement de l'*objet* automate. Bien que ces problèmes soient résolus, on peut encore se demander s'il est possible de "lire" sur l'automate les propriétés recherchées. Les travaux que j'ai menés consistent donc plutôt à étudier des morphismes *entre* automates, et à exprimer dans leur structure même les propriétés que l'on cherche à démontrer.

Un premier sujet d'étude est la *dérivation des expressions rationnelles*, que j'ai étendu aux expressions avec multiplicité. Elle s'est avérée être une porte ouverte sur l'étude des relations entre automates et expressions.

Le théorème de Kleene établit que les automates finis et les expressions rationnelles ont le même pouvoir d'expression. La preuve consiste à exhiber des algorithmes permettant le passage d'un mode de représentation à un autre. Toutefois, parmi les automates qui reconnaissent un même langage, il peut y avoir des différences considérables (*enlacement*, ambiguïté, *etc.*), même si nous verrons par ailleurs qu'on peut toujours passer d'un automate à un autre équivalent par des fusions et dédoublements d'états *ad hoc*. De même, en toute généralité, les expressions rationnelles équivalentes sont fort différentes, puisqu'il a été montré par D. Krob [27] que la réécriture d'une expression en une autre nécessite un nombre de règles non borné.

La question que l'on se pose est donc la suivante : peut-on trouver des invariants plus précis que le langage reconnu pour les algorithmes classiques de transformation. Il est par exemple prouvé (*cf.* [52, 27]) que les différentes expressions calculables à partir d'un même automate par l'algorithme d'élimination sont équivalentes *modulo* un nombre fini de règles de réécriture.

Une autre façon d'aborder le problème est de voir dans quelle mesure ces algorithmes sont inversibles, c'est-à-dire quelle est la trace dans l'expression calculée de l'automate de départ ou inversement. P. Caron et D. Ziadi [10] ont montré qu'à partir de l'automate standard d'une expression, on peut retrouver l'expression elle-même. Je me suis intéressé au problème inverse : savoir dans quelle mesure une expression rationnelle calculée à partir d'un automate garde la trace de cet automate [42, 44]. Au delà de l'intérêt purement théorique, cette question a un intérêt pratique ; en effet, la taille d'une expression rationnelle peut être exponentielle par rapport à celle de l'automate à partir duquel elle est calculée. D'autre part, le calcul d'un automate à partir d'une expression rationnelle quelconque donne un automate qui a un nombre d'états linéaire et un nombre de transitions quadratique (on peut en fait améliorer ces bornes et obtenir un objet de taille $O(n \log^2 n)$ [23, 19]). Bref, si une expression est issue d'un automate et qu'on retrouve la trace de celui-ci, ceci permet de construire un objet exponentiellement plus petit.

Si une expression est calculée à partir d'un automate, nous proposons l'algorithme suivant pour retrouver celui-ci : d'abord construire un automate des *dérivées* proche de celui défini par V. Antimirov [1], puis en calculer le *co-quotient* minimal. Cet algorithme ne produit pas toujours le résultat souhaité. Si l'automate de départ admet un co-quotient propre, on risque de trouver un automate plus petit, mais nous avons aussi décrit certaines configurations qui peuvent conduire à un mauvais résultat, même sur un co-quotient minimal. Cependant, on a montré que notre procédure fonctionne pour un automate co-déterministe minimal. En fait, le caractère co-déterministe de l'automate est conservé par le calcul de l'expression suivi de la dérivation. De même, si une expression est calculée à partir d'un automate déterministe, on peut obtenir un automate déterministe équivalent en lui appliquant des dérivations à droite, ce qui évite une étape de déterminisation potentiellement exponentielle.

Un autre aspect de l'étude des expressions rationnelles concerne la hauteur d'étoile. Ce problème, posé par L.C. Eggan [15] en 1963, ne fut résolu qu'en 1988 par K. Hashiguchi [21]. Plus récemment, en 2004, D. Kirsten [24] en a donné une preuve algébrique très élégante.
Pour ma part, je me suis intéressé à la conjecture suivante : l'automate universel d'un langage rationnel contient un sous-automate reconnaissant le langage et correspondant à une expression de hauteur d'étoile minimale. L.C. Eggan avait en effet montré le lien entre l'*enlacement* d'un automate et la hauteur d'étoile.
L'automate universel d'un langage est un automate, inspiré par J.H. Conway [14] tel que tout automate qui reconnaît le langage admet une image par morphisme dans cet automate.

J'ai montré que dans le cas des langages réversibles la conjecture est non seulement vraie, mais on peut en plus trouver un tel sous-automate qui est quasi-réversible. On le voit, dans ce cas, cette approche donne en plus des informations sur la forme du résultat.

Avec Marie-Pierre Béal et Jacques Sakarovitch, nous avons étudié la notion de conjugaison d'automates [4]. Cette notion, bien connue dans le cadres des systèmes dynamiques, peut se transposer aux automates finis avec multiplicité et s'avère particulièrement fructueuse. Nous avons montré, dans le cadre des automates à multiplicité dans un corps, $\mathbb{Z}$, $\mathbb{N}$ ou $\mathbb{B}$ que, d'une part, toute paire d'automates équivalents est reliée par au plus deux conjugaisons et, d'autre part, toute conjugaison se décompose en un *revêtement* et un *co-revêtement*. Ceci entraîne finalement que ces fusions et éclatements d'états permettent de passer d'un automate donné à n'importe quel autre équivalent.

D'un autre côté, j'ai étudié avec Ines Klimann, Jean Mairesse et Christophe Prieur la déterminisation des automates max-plus [25, 26]. La question est de savoir si une série réalisée par un automate max-plus peut aussi être réalisée par un automate max-plus déterministe ; cette série est alors dite séquentielle. Ce problème reste encore ouvert. M. Mohri [47] a montré que l'on peut décider la séquentialité des séries réalisées par des automates non ambigus. Nous avons montré qu'étant donné un automate dont le nombre de chemins étiquetés par un mot donné est borné indépendamment du mot (on dit que l'automate est finiment ambigu), on peut décider si la série correspondante est réalisable par automate non ambigu, ce qui étend le cadre dans lequel la séquentialité est décidable.

Pour cela, nous montrons qu'un automate finiment ambigu est décomposable en union finie d'automates non ambigus. Ensuite, en comparant les poids des chemins réussis (et notamment de leurs boucles) étiquetés par le même mot dans les divers automates de cette union, on donne un critère de non ambiguïté.

Dans le même domaine, j'ai étudié avec Jean Mairesse les séries rationnelles réalisables à la fois par automate max-plus et automates min-plus [36]. Nous avons montré que ces séries sont exactement celles réalisables par automates non ambigus, ce qui unifie deux familles de séries sur lesquels la plupart des problèmes classiques (équivalence, séquentialité) sont décidables. Pour cela, nous avons ré-exprimés un certains nombres de résultats connus dans le cadre des automates max-plus à multiplicités réelles : propriété de Fatou (si la série réalisée est négative, on peut faire de sorte à n'avoir que des poids négatifs), décider si une série est négative, si elle a un coefficient positif, *etc.*

Une autre part de l'activité de ces dernières années a été le co-encadrement du projet Vaucanson [37, 38, 13, 53]. Ce projet, mené avec des étudiants du LRDE à l'Épita, a pour objectif de construire une bibliothèque C++ générique de manipulation d'automates avec multiplicité. La particularité de cette bibliothèque est de pouvoir accepter des multiplicités dans n'importe quel semi-anneau. Outre l'encadrement et le support de connaissances, j'ai aussi implanté quelques algorithmes dont certains sont inédits : dérivations avec multiplicité, calcul du quotient minimal d'un automate à multiplicité, *etc.* Je présenterai ces algorithmes et la façon dont ils sont implantés.

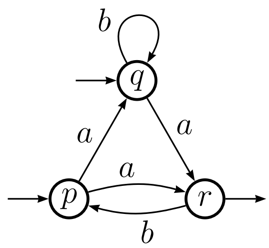
Chapitre 1

Définitions

Voici une brève description des objets étudiés dans ce mémoire. Évidemment, pour une vision plus approfondie des notions présentées dans ce chapitre, il est fortement conseillé de se reporter aux ouvrages de références ([16, 52] par exemple).

1 Automates

Un automate (fini) est constitué d'un ensemble fini d'*états* (représentés habituellement par des ronds) et d'un ensemble fini de *transitions* (représentées par des flèches). Chaque transition part d'un état et arrive à un autre état (éventuellement le même) ; elle porte en outre une *étiquette*, qui est une lettre choisie dans un ensemble fini appelé *alphabet*. De plus, certains états sont *initiaux* (ce qui est symbolisé par une petite flèche pointant sur le rond) ou *terminaux* (avec une petite flèche sortante).



De façon formelle, un automate est donc caractérisé par un quintuplet (Q, A, E, I, T), où Q est l'ensemble des états, A l'alphabet, E l'ensemble des transitions, et I et T les ensembles des états initiaux et terminaux respectivement. Les ensembles I et T sont évidemment inclus dans Q et toute transition e de E sera désignée par un triplet (p, a, q), où p et q sont les états de départ et d'arrivée de la transition et a son étiquette.

Considérons un automate. Un *chemin* est une suite de transitions successives, c'est-à-dire que chaque transition de la suite a pour état de départ l'état d'arrivée de la transition précédente. La longueur d'un chemin est le nombre de transitions qui le constituent. Un chemin est *réussi*, s'il part d'un état initial et arrive dans un état final. L'étiquette d'un chemin est le mot obtenu en concaténant les étiquettes des transitions qui forment le chemin. Un mot est *accepté* par l'automate s'il est l'étiquette d'un chemin réussi. Remarquons que si un état est à la fois initial et terminal, il existe un chemin de longueur nulle dont l'étiquette est un mot de longueur nulle appelé *mot vide* et noté 1_{A^*} ou ε. Le *langage reconnu* par un automate est l'ensemble des mots qu'il accepte. Un ensemble de mots sur un alphabet donné est dit *reconnaissable* s'il est reconnu par un automate fini.

Un *morphisme* d'un automate $\mathcal{A}$ dans un automate $\mathcal{B}$ est une application φ des états de $\mathcal{A}$ dans ceux de $\mathcal{B}$ telle que pour toute transition (p, a, q) de $\mathcal{A}$, $(\varphi(p), a, \varphi(q))$ est une transition de $\mathcal{B}$, et pour tout état initial (*resp.* final) p, l'état $\varphi(p)$ est initial (*resp.* final).

2 Expressions rationnelles

Une expression rationnelle sur un alphabet A est une formule finie égale à 0, 1, a (avec $a \in A$), $(\mathsf{E}+\mathsf{F})$, $(\mathsf{E}\cdot\mathsf{F})$ ou $(\mathsf{E})^*$, où E et F sont des expressions rationnelles.

On peut appliquer les règles de priorité suivantes : '$*$' prioritaire sur '$\cdot$', lui-même prioritaire sur '$+$', application des opérateurs de même priorité de la gauche vers la droite. On peut alors supprimer les parenthèses superflues. De même, on n'écrira pas toujours le symbole '$\cdot$'.

L'ensemble des mots sur un alphabet A, noté A^*, est muni d'une multiplication naturelle, qui consiste à concaténer les mots. Le mot vide est l'élément neutre de cette opération. On peut étendre cette multiplication aux ensembles de mots :

$$\forall X, Y \subseteq A^*,\ X.Y = \bigcup_{u \in X, v \in Y} u.v.$$

La puissance n-ième d'un ensemble X de mot est récursivement définie par $X^0 = 1_{A^*}$ et $X^{n+1} = X^n.X$.

On peut désormais donner une *interprétation* des expressions rationnelles : $|0| = \emptyset$, $|1| = 1_{A^*}$, $|a| = \{a\}$ (si $a \in A$), et si E et F sont des expressions rationnelles, $|(\mathsf{E}+\mathsf{F})| = |\mathsf{E}| \cup |\mathsf{F}|$, $|(\mathsf{E}\cdot\mathsf{F})| = |\mathsf{E}|.|\mathsf{F}|$, et $|(\mathsf{E})^*| = \bigcup_{n\in\mathbb{N}} |\mathsf{E}|^n$.

Un ensemble de mots est *rationnel* s'il est l'interprétation d'une expression rationnelle. La plupart du temps, on confondra l'expression avec son interprétation, sauf si l'on veut étudier les expressions elles-mêmes.

Le théorème de Kleene établit que dans le cas des mots sur un alphabet fini, les ensembles reconnaissables et rationnels sont les mêmes.

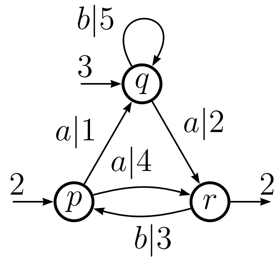
3 Automates avec multiplicité

Un *semi-anneau* $\mathbb{K}$ est un ensemble muni d'une multiplication et d'une addition ; ces deux opérations sont associatives et $\mathbb{K}$ contient un élément neutre pour chacune : $0_{\mathbb{K}}$ pour l'addition et $1_{\mathbb{K}}$ pour la multiplication ; de plus, l'addition est commutative, la multiplication distributive sur l'addition et $0_{\mathbb{K}}$ est absorbant pour la multiplication.

Si en plus l'addition est idempotente ($\forall k \in \mathbb{K}, k + k = k$), on dira que le semi-anneau est idempotent.

Tout anneau est évidemment un semi-anneau, mais il existe de nombreux autres exemples : semi-anneau de Boole ($\{\mathsf{Vrai},\mathsf{Faux}\}$,$\mathsf{ou}$,$\mathsf{et}$), entiers naturels, *etc.*

Les automates à multiplicité sont une extension de la notion d'automate. Un automate à multiplicité sur un semi-anneau $\mathbb{K}$ ou $\mathbb{K}$-automate est un automate dans lequel chaque transition porte non seulement une étiquette, mais aussi un poids appartenant à $\mathbb{K}$. Toute flèche initiale (*resp.* terminale) porte elle aussi un poids. Un automate à multiplicité est donc caractérisé par un sextuplet $(Q, A, \mathbb{K}, E, I, T)$, où une transition e de E est de la forme (p, a, k, q), un élément de I de la forme (k, p) et un élément de T de la forme (p, k). Pour un couple d'états (p, q) fixé et une lettre a donnée, il y a au plus qu'une transition de p à q étiquetée par a. De même, chaque état n'a au plus qu'une flèche initiale et une flèche finale.



Le poids d'un chemin est obtenu en multipliant les poids des transitions du chemin : le poids d'un chemin réussi est obtenu en multipliant le poids du chemin à gauche par le poids de la flèche initiale et à droite par celui de la flèche terminale.

La multiplicité d'un mot dans l'automate est la somme des poids des chemins réussis étiquetés par ce mot. Ainsi, étant donné un $\mathbb{K}$-automate, à chaque mot est associé un élément de $\mathbb{K}$: le $\mathbb{K}$-automate *réalise* une fonction de A^* dans $\mathbb{K}$ (qu'on peut voir comme une série sur A^* à coefficients dans $\mathbb{K}$). On notera $\langle s, w\rangle$ le coefficient de w dans la série s.

Remarquons que fixer le poids d'une transition ou d'une flèche initiale ou terminale à $0_{\mathbb{K}}$ revient à ne pas considérer cette transition ou flèche. Inversement, s'il n'y a pas de transition entre deux états, on peut toujours supposer qu'il en existe une de poids $0_{\mathbb{K}}$. De même, on dira qu'un état est initial ou terminal, si le poids correspondant est non nul.

$\mathbb{K}$-représentation Soit $\mathcal{A}=(Q, A, \mathbb{K}, E, I, T)$ un automate à multiplicité. On définit un vecteur ligne λ et un vecteur colonne ν de taille Q, ainsi qu'une application μ de A dans les matrices carré de taille Q de la façon suivante :

$$\lambda_p = \sum_{(k,p)\in I} k, \quad \nu_p = \sum_{(p,k)\in T} k \quad \text{et} \quad \mu(a)_{p,q} = \sum_{(p,a,k,q)\in E} k.$$

Le triplet (λ, μ, ν) est une $\mathbb{K}$-représentation (linéaire) de la série réalisée par $\mathcal{A}$: le poids du mot $w = w_1 ... w_k$ est $\lambda\mu(w_1)...\mu(w_k)\nu$.

On identifiera I à λ, T à ν et E à la matrice de transition $M = \sum_{a\in A}\mu(a)a$. Pour désigner un $\mathbb{K}$-automate, on utilisera donc indifféremment (I, M, T) ou (I, μ, T), la minuscule grecque correspondant à la $\mathbb{K}$-représentation et la majuscule à la matrice de transition.

$$I = \begin{bmatrix} 2 & 3 & 0 \end{bmatrix}, M = \begin{bmatrix} 0 & a & 4a \\ 0 & 5b & 2a \\ 3b & 0 & 0 \end{bmatrix} T = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

Chapitre 2

Conjugaisons

La conjugaison des automates que nous étudions ici est intéressante à plusieurs titres. Elle est directement inspirée d'une notion courante en symbolique dynamique ; elle permet d'exprimer un ensemble de transformations sous une forme commune et enfin, elle permet d'expliciter les liens qui existent entre l'équivalence des séries rationnelles et les revêtements d'automates.

Ce travail récent [4] réserve encore probablement des développements.

Definition 2.1 *Soit $\mathcal{A} = (I, M, T)$ et $\mathcal{B} = (J, N, U)$ deux $\mathbb{K}$-automates. On dit que $\mathcal{A}$ est conjugué à $\mathcal{B}$ par X s'il existe une matrice X telle que*

$$IX = J, \quad MX = XN \quad \text{et} \quad T = XU.$$

On note alors $\mathcal{A} \stackrel{X}{\Longrightarrow} \mathcal{B}$.

Cette relation est transitive, réflexive, mais pas symétrique ; il s'agit d'un pré-ordre.

Notons qu'on ne fait absolument aucune supposition *a priori* sur la matrice X. Celle-ci peut être dégénérée et même nulle ! Contrairement aux systèmes dynamiques, la présence des vecteurs initiaux et finaux garantit malgré tout l'équivalence des automates.

Il est immédiat que deux automates conjugués sont équivalents. Il est a peu près aussi immédiat que deux automates équivalents ne sont pas forcément conjugués.

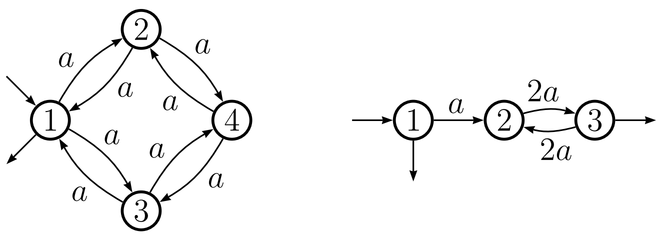


Fig. 2.1- Deux $\mathbb{N}$-automates équivalents non conjugués

Le premier résultat que je vais présenter montre que toutefois, étant donnés deux $\mathbb{K}$-automates équivalents, il existe toujours un $\mathbb{K}$-automate conjugué aux deux. Ceci est vrai pour de nombreux semi-anneaux $\mathbb{K}$: $\mathbb{B}$, $\mathbb{N}$, $\mathbb{Z}$, corps, *etc.*

Le second résultat donne une interprétation structurelle de la conjugaison. On peut définir sur les $\mathbb{K}$-automates des opérations élémentaires (fusion ou éclatement d'états, circulation de coefficients) qui, si les dites opérations sont bien définies, préservent la série réalisée. On montre que toute conjugaison s'exprime comme un éclatement d'états suivi d'une circulation et d'une fusion d'états.

Ces deux résultats permettent de répondre à une question que je me posais depuis un certain temps. Peut-on grâce à ces éclatements et circulation passer d'un automate à n'importe quel autre automate équivalent ? La réponse, surprenante, est positive.

1 Équivalence et conjugaison

Proposition 2.1 *Dans cet énoncé, $\mathbb{K}$ est égal à $\mathbb{N}$, $\mathbb{Z}$ ou est un corps.*
Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates (resp. deux $\mathbb{K}$-automates ou deux transducteurs fonctionnels*) équivalents. Il existe un automate $\mathcal{C}$ (resp. un $\mathbb{K}$-automate ou un transducteur) conjugué à $\mathcal{A}$ et $\mathcal{B}$.*

Nous allons brièvement esquisser la preuve de ce résultat dans les différents cadres cités ainsi que les constructions sous-jacentes.

1.1 Le cas booléen

La preuve du cas booléen s'appuie sur une construction bien connue : la déterminisation.

Rappelons en deux mots de quoi il s'agit. Étant donné un automate $\mathcal{A} = (I, \mu, T)$, on calcule l'ensemble des vecteurs $I\mu(w)$ (qui représentent des sous-ensembles) qui forment les états de l'automate déterminisé $\mathcal{D}$; I est l'état initial, le successeur d'un état α par la lettre a est l'état $\alpha\mu(a)$, et l'état α est final si et seulement si $\alpha T = 1$.

Considérons maintenant la matrice X dont les lignes sont les états de $\mathcal{D}$. On vérifie immédiatement que $\mathcal{D}$ est conjugué à $\mathcal{A}$ par X.

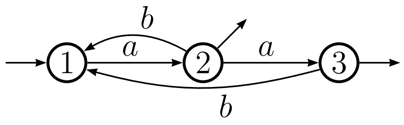
Pour montrer la proposition 2.1, il suffit maintenant de considérer le déterminisé de l'union de $\mathcal{A}$ et $\mathcal{B}$. Chaque état est un vecteur formé de deux parties $[\alpha|\beta]$, α étant indicé par les états de $\mathcal{A}$ et β par ceux de $\mathcal{B}$. Notons qu'un état est final si et seulement si $[\alpha|\beta]\left[\frac{T}{U}\right] = 1$, ce qui est équivalent, puisque $\mathcal{A}$ et $\mathcal{B}$ sont équivalents, à $\alpha T = \beta U = 1$.

Si l'on considère la matrice dont les lignes sont les états du déterminisé, elle se décompose en $[X|Y]$, les colonnes de X étant indicées par les états de $\mathcal{A}$ et celles de Y par ceux de $\mathcal{B}$ et on obtient $\mathcal{D} \stackrel{X}{\Longrightarrow} \mathcal{A}$ et $\mathcal{D} \stackrel{Y}{\Longrightarrow} \mathcal{B}$.

Exemple. Considérons les deux automates ci-dessous.

Fig. 2.2- Deux automates équivalents

Le déterminisé commun est l'automate ci-dessous dont les états correspondent aux lignes de la matrice ci-contre :



$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

Cet automate est conjugué aux deux précédents et les matrices de conjugaison sont les deux moitiés de la matrice ci-dessus.

1.2 Le cas des entiers naturels

Soit $\mathcal{A} = (I, \mu, T)$ et $\mathcal{B} = (J, \nu, U)$ deux $\mathbb{N}$-automates équivalents. Dans le cas des entiers naturels, on le sait, on ne peut pas toujours calculer un automate déterminisé pour la simple raison que l'ensemble des vecteurs $[I\mu(w)|J\nu(w)]$ n'est pas toujours fini. Par contre, on peut calculer un ensemble fini de vecteurs $[\alpha|\beta]$ qui engendrent un semi-module contenant les $[I\mu(w)|J\nu(w)]$ et tels que, pour tout mot w, $\alpha\mu(w)T = \beta\nu(w)U$.

Pseudo-réduction conjointe gauche de $\mathcal{A} = (I, \mu, T)$ et $\mathcal{B} = (J, \nu, U)$

> S : liste vide
> P : pile initialisée à $\{1_{A^*}, (I : J)\}$
> **tant que** $P \neq \emptyset$
> $(w, \alpha) :=$pop(P)
>   **pour tout** β dans S
>   **tant que** $\beta \leq \alpha$
> $\alpha := \alpha - \beta$
>   **si** $\alpha \neq 0$
> $S \leftarrow \alpha$
>   **pour tout** a dans A
> $P \leftarrow (wa, \alpha(\mu(a) : \nu(a)))$
> **retourner** S

La procédure termine grâce au fait que l'ordre produit sur $\mathbb{N}^k$ est un bon ordre partiel et que tout sous-ensemble d'éléments deux à deux incomparables est fini.

La liste S qu'on construit ainsi est un ensemble générateur (qui n'est évidemment pas une base) du semi-module contenant les $[I\mu(w)|J\nu(w)]$. On peut ne conserver de cette liste que les vecteurs minimaux.

Pour tout vecteur $[\alpha_i|\beta_i]$ de S, pour toute lettre a, il existe des coefficients $\xi(a)_{i,j}$, et des coefficients $\mathbb{K}_j$ tels que

$$[\alpha\mu(a)|\beta\nu(a)] = \sum_{[\alpha_j|\beta_j]\in S} \xi(a)_{i,j}[\alpha_j|\beta_j], \quad [I|J] = \sum_{[\alpha_j|\beta_j]\in S} K_j[\alpha_j|\beta_j],$$

et on pose $V_i = \alpha_i T = \beta_i U$.

L'automate $\mathcal{C} = (K, \xi, V)$ est conjugué aux automates $\mathcal{A}$ et $\mathcal{B}$. Pour obtenir les matrices de conjugaison, il suffit de prendre respectivement la matrice formée des α_i ou des β_i.

1.3 Le cas des entiers et des corps (gauche)

Ce cas est en réalité très similaire au cas précédent. La différence est que étant donné la structure des $\mathbb{Z}$-modules et des espaces vectoriels, on peut non seulement calculer un ensemble générateur du module ou du corps engendré par les $[I\mu(w)|J\nu(w)]$ mais une base :

Réduction gauche conjointe dans un corps de $\mathcal{A} = (I, \mu, T)$ et $\mathcal{B} = (J, \nu, U)$
    S : liste vide
    P : pile initialisée à $\{1_{A^*}, (I : J)\}$
    tant que $P \neq \emptyset$
        $(w, \alpha) := \text{pop}(P)$
        pour tout $\beta \in S$
            $j := \mathsf{min}\{l \mid \beta_l \neq 0\}$
            $h := \mathsf{min}\{l \mid \alpha_l \neq 0\}$
            si $(h < j)$
            alors break
            sinon $\alpha := \alpha - \alpha_j \beta_j^{-1} \beta$
        si $(\alpha \neq 0)$ **alors**
            insère α dans S
            pour tout $a \in A$
                $P \leftarrow (aw, \alpha(\mu(a) : \nu(a))))$
    retourner S

On le voit, le calcul consiste à maintenir à jour une base (échelonnée pour des raisons de facilités). Tout vecteur incorporé à la base l'est définitivement, en vertu du théorème de la base incomplète. Ce théorème n'a plus cours dans $\mathbb{Z}$, aussi faut-il adapter l'algorithme.

Réduction gauche conjointe dans $\mathbb{Z}$ de $\mathcal{A} = (I, \mu, T)$ et $\mathcal{B} = (J, \nu, U)$
    S : liste vide
    P : pile initialisée à $\{1_{A^*}, (I : J)\}$
    while $(P \neq \emptyset)$
        $(w, \alpha) := \text{pop}(P)$
        pour tout $\beta \in S$
            $j := \mathsf{min}\{l \mid \beta_l \neq 0\}$
            $h := \mathsf{min}\{l \mid \alpha_l \neq 0\}$
            si $(h < j)$
            alors break
            sinon si $(h = j)$ **alors**
                Soit (p, q) tel que $p\alpha_j + q\beta_j = \text{gcd}(\alpha_j, \beta_j)$ avec $q \neq 0$
                $\gamma := p\alpha + q\beta$
                $\alpha := (\beta_j \alpha - \alpha_j \beta)/\gamma_j$
                $\beta := \gamma$
            sinon continue
        si $(\alpha \neq 0)$ **alors**
            insère α dans S
            pour tout $a \in A$
                $P \leftarrow (aw, \mu(a)\alpha)$
    retourner S

On peut revenir un instant sur le nom de Réduction gauche conjointe donné à ces algorithmes. Cette procédure est "conjointe" parce qu'elle s'applique à l'union de deux automates, "gauche" parce que l'on peut effectuer un algorithme similaire ("réduction droite") pour calculer une base du module ou de l'espace vectoriel engendré par les $[\mu(a)T, \nu(a)U]$.

Enfin, il s'agit d'une "réduction" car la réduction gauche suivie de la réduction droite donne un $\mathbb{K}$-automate avec un nombre d'état minimal parmi tous les automates équivalents (*cf.* [7, 52]).

Exemple. Considérons les deux $\mathbb{Z}$-automates $\mathcal{A}_5$ et $\mathcal{B}_5$ équivalents ci-dessous :

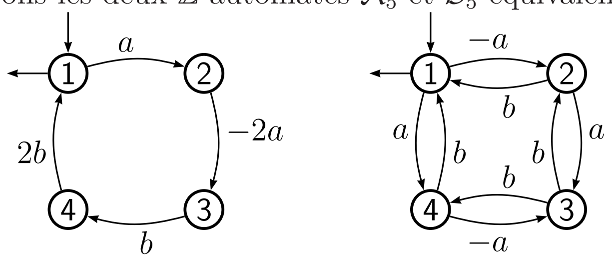
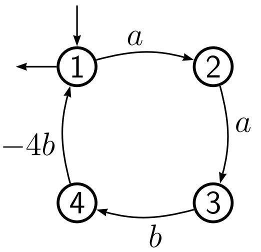


Fig. 2.3- Deux $\mathbb{Z}$-automates équivalents.

Leur réduction gauche conjointe donne un $\mathbb{Z}$-automate dont les états correspondent à la matrice présentée ci-contre :



$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -2 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & -2 & 0 & -2 & 0 & -2 \end{array}\right]$$

Cet automate est conjugué à $\mathcal{A}_5$ et $\mathcal{B}_5$; on voit qu'en ce qui concerne$\mathcal{A}_5$, il s'agit seulement d'une circulation de coefficients (matrice diagonale). En fait, les automates $\mathcal{A}_5$ et $\mathcal{B}_5$ sont directement conjugués. En effet, la matrice de circulation est inversible dans $\mathbb{Q}$ et son produit avec l'autre matrice de conjugaison donne une matrice à coefficients entiers qui est une matrice de conjugaison entre ces deux automates.

1.4 Transducteurs fonctionnels

Un transducteur dont les sorties sont des mots de B^* peut être vu comme un automate à multiplicité dans $\mathsf{Rat}(B^*)$. La notion de conjugaison est donc bien définie pour les transducteurs et on peut montrer la proposition 2.1.

Contrairement aux semi-anneaux traités dans les paragraphes précédents, on ne dispose pas pour $\mathsf{Rat}(B^*)$ d'opération de soustraction. Si on veut obtenir un ensemble fini de $[\alpha_i|\beta_i]$ générateur des $[I\mu(w), J\nu(w)]$ tout en respectant $\alpha_i\mu(w)T = \beta_i\nu(w)U$ pour tout w, il faudra trouver d'autres critères pour "réduire" les $[I\mu(w), J\nu(w)]$.

C'est là que l'hypothèse de fonctionnalité des transducteurs va nous servir. L'idée est la suivante : si un mot mène dans deux états avec des sorties trop différentes (soit parce qu'elles ont des préfixes incomparables, soit parce qu'elles sont de longueurs très différentes), les mots qu'on peut reconnaître en partant de ces états seront forcément différents.

Rappelons d'abord l'algorithme classique de séquentialisation. On utilise deux opérations de base sur les vecteur de mot : si α est un tel vecteur, $\breve{\alpha}$ est le plus long préfixe

commun des coordonnées non nulles de α et $\bar{\alpha} = \check{\alpha}^{-1}\alpha$. À partir d'un transducteur $\mathcal{T} = (I, \mu, T)$, on construit incrémentalement le transducteur $\mathcal{S}$ de la façon suivante :
– $\bar{I}$ est l'état initial, avec $\check{I}$ comme poids,
– de l'état α, en lisant la lettre a, on va dans l'état $\bar{\beta}$ en sortant $\check{\beta}$, avec $\beta = \alpha\mu(a)$.
– l'état α est final si $u = \alpha T$ n'est pas nul ; son poids terminal est alors u.

Évidemment, si le transducteur n'est pas séquentialisable, cet algorithme construit un transducteur infini. Modifions le donc.

On ne veut garder dans un même vecteur que des mots “pas trop différents”.

Dans l'algorithme de séquentialisation, chaque vecteur obtenu sera décomposé en une somme de vecteurs à support disjoints tels que :
– chaque vecteur contient un élément minimum pour l'ordre préfixe ;
– si w est un coefficient non nul et non minimum d'un vecteur, il existe parmi les coefficients du vecteur un mot de longueur au moins $|w| - K$ préfixe propre de w.

La constante K dépend du nombre d'états du transducteur et de la taille des mots sortis sur chaque transition.

On peut s'interroger, à juste titre, sur l'intérêt de cette construction, puisqu'on ne peut pas décrire de propriété particulière du résultat, à part qu'il s'agit d'un transducteur non ambigu. Notons quand même que si le transducteur est séquentialisable, on obtient effectivement un transducteur séquentiel.

L'intérêt de la construction se révèle lorsqu'on applique l'algorithme à l'union de deux transducteurs fonctionnels $\mathcal{T}$ et $\mathcal{U}$ équivalents ; on obtient comme dans les paragraphes précédents un transducteur conjugué à la fois à $\mathcal{T}$ et à $\mathcal{U}$.

1.5 Autres semi-anneaux

L'énumération des semi-anneaux dans lesquels la proposition 2.1 est vérifiée donne l'impression que l'on peut sans trop de difficultés la décliner à l'envi dans n'importe quel semi-anneau.

Pour autant que je sache, ce n'est pas le cas. Par exemple, dans le cas des semi-anneaux tropicaux, les arguments déployés dans les paragraphes précédents ne s'appliquent pas ; il en va de même, *a fortiori* pour les transducteurs non fonctionnels.

2 Revêtements et conjugaisons

On montre dans cette partie que toute conjugaison peut se décrire en terme d'éclatements et de fusion d'états. On a aussi besoin de faire “circuler” les coefficients dans l'automate.

2.1 Circulation de coefficients

Faire circuler un coefficient c à travers un état d'un $\mathbb{K}$-automate consiste à *diviser* les poids des transitions entrantes sur cet état par c et à multiplier les poids des transitions sortantes par c. Ceci ne change donc en rien le graphe sous-jacent à l'automate.

En fait, faire circuler les poids c_p à travers les états p de l'automate $\mathcal{A} = (I, M, T)$ consiste à construire l'automate $\mathcal{B} = (ID^{-1}, DMD^{-1}, DT)$, où D est la matrice diagonale des c_p.

Cette écriture suppose D inversible et on pourrait plus simplement dire que le résultat $\mathcal{B} = (J, N, U)$ de la circulation vérifie :

$$I = JD, \quad DM = ND \quad \text{et} \quad DT = U,$$

ce qui est bien équivalent si D est inversible. Attention toutefois, car si D n'est pas inversible, on n'est pas assuré de l'existence d'une solution $\mathcal{B}$ à cette équation ou de son unicité. On supposera donc dans la suite que la matrice de circulation est inversible.

Remarquons pour finir que la circulation de coefficients est une conjugaison.

2.2 $\mathbb{K}$-revêtements et $\mathbb{K}$-quotients

Le but de ce paragraphe est de définir des éclatements ou fusions d'états dont on assure, grâce à des critères *locaux*, qu'ils préservent la série réalisée. Une étude complète de ces transformations est donnée en [52].

Formellement, un automate $\mathcal{B}$ est obtenu à partir de $\mathcal{A}$ par une fusion d'états si il existe une application φ surjective des états de $\mathcal{A}$ sur ceux de $\mathcal{B}$. Ceci ne précise pas comment sont définies les transitions dans $\mathcal{B}$ à partir de celles de $\mathcal{A}$. Dans le cas des automates classiques, cette fusion correspond à un morphisme surjectif et on définira dans $\mathcal{B}$ une transition (r, a, s) si et seulement si il existe (p, q) dans $\varphi^{-1}(r) \times \varphi^{-1}(s)$ tel que (p, a, q) est une transition de $\mathcal{A}$. De telles fusions ne garantissent pas la conservation du langage reconnu. C'est pourquoi nous allons ajouter des contraintes sur les états que l'on peut fusionner et définir les transitions dans le résultat de façon moins brutale.

Un $\mathbb{K}$-quotient est le résultat d'une fusion entre états qui ont les même successeurs (*modulo* la fusion). Il faut faire un peut attention à ce qu'on entend par là dans le cadre des $\mathbb{K}$-automates. “Avoir les mêmes successeurs” ne signifie pas seulement aller dans les mêmes états, mais y aller avec le même poids !

Ainsi, si φ est une fusion d'états de l'automate $\mathcal{A} = (I, M, T)$, c'est un $\mathbb{K}$-quotient si et seulement si :

$$p\varphi = q\varphi \Longrightarrow \quad T_p = T_q \quad \text{et} \quad \forall r, \sum_{s \in \varphi^{-1}\varphi(r)} M_{p,s} = \sum_{s \in \varphi^{-1}\varphi(r)} M_{q,s}. \tag{2.1}$$

L'automate résultant de la fusion est alors défini ainsi. Le poids initial d'un état est la somme des poids initiaux des états fusionnés, son poids final est le poids final commun à tous les états fusionnés et le poids d'une transition étiquetée par a entre r et s est la somme, pour n'importe quel état p dans $\varphi^{-1}(r)$ des transitions menant de p aux états de $\varphi^{-1}(s)$ étiquetées par a.

Ceci s'exprime de manière remarquablement simple et concise avec des matrices. En effet, si $\mathcal{A} = (I, M, T)$ est un $\mathbb{K}$-automate et si φ est une fusion d'états, alors φ est un $\mathbb{K}$-quotient si et seulement si il existe un $\mathbb{K}$-automate $\mathcal{B} = (J, N, U)$ tel que

$$IX_\varphi = J, \quad MX_\varphi = X_\varphi N \quad \text{et} \quad T = X_\varphi U,$$

où X_φ est la matrice de l'application φ. L'automate $\mathcal{B} = (J, N, U)$ est alors un $\mathbb{K}$-quotient de $\mathcal{A}$.

Le revêtement est l'inverse du quotient ; dans ce qui précède, $\mathcal{A}$ est un $\mathbb{K}$-revêtement de $\mathcal{B}$. L'équation ci-dessus est exactement celle d'une conjugaison, donc si $\mathcal{A}$ est un $\mathbb{K}$-revêtement de $\mathcal{B}$. $\mathcal{A}$ est conjugué à $\mathcal{B}$.

Enfin, étant donné un $\mathbb{K}$-automate $\mathcal{A}$ et une fusion φ de ses états, si cette fusion est effectivement un $\mathbb{K}$-quotient, le résultat $\mathcal{B}$ est unique. Par contre, étant donné un $\mathbb{K}$-automate $\mathcal{B}$ et un éclatement φ^{-1} de ses états, il y a *a priori* plusieurs $\mathbb{K}$-automates $\mathcal{A}$ qui sont des $\mathbb{K}$-revêtements de $\mathcal{B}$ selon φ.

En effet, la seule contrainte pour définir I et M est :

$$\forall s, \sum_{q\in\varphi^{-1}(s)} I_q = J_s \quad \text{et} \quad \forall p, \sum_{q\in\varphi^{-1}(s)} M_{p,q} = N_{\varphi(p),s};$$

le vecteur T, quant à lui, est imposé : $T = X_\varphi U$.

Un $\mathbb{K}$-co-quotient est une opération similaire, sauf qu'elle consiste à fusionner les états qui ont les mêmes prédécesseurs ; elle se résume par l'équation suivante :

$$I = J^t X_\varphi, \quad {}^t X_\varphi M = N^t X_\varphi \quad \text{et} \quad {}^t X_\varphi T = U.$$

2.3 Énoncé

Nous avons maintenant les outils nécessaire pour exposer le résultat :

Proposition 2.2 $\mathbb{K} = \mathbb{N}, \mathbb{Z}$ *ou un corps.*
Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates (resp. deux $\mathbb{K}$-automates ou deux transducteurs fonctionnels*) émondés tels que $\mathcal{A}$ est conjugué à $\mathcal{B}$ par X. Alors, il existe deux automates (resp. deux $\mathbb{K}$-automates ou deux transducteurs* fonctionnels*) $\mathcal{A}'$ et $\mathcal{B}'$ égaux à une circulation de coefficients près, tels que $\mathcal{A}'$ est un co-revêtement de $\mathcal{A}$ et $\mathcal{B}'$ un revêtement de $\mathcal{B}$.*

2.4 Le cas booléen

Dans le cas booléen, la seule circulation de coefficients est l'identité, on n'en tient donc pas compte. Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates émondés d'ensembles d'états respectifs Q et R tels que $\mathcal{A} \overset{X}{\Longrightarrow} \mathcal{B}$. Numérotons de 1 à k les coefficients non nuls de X ; (i_r, j_r) est la coordonnée du coefficient numéro r. La matrice H_φ de taille $k \times Q$ (*resp.* H_ψ de taille $k \times R$) a des coefficients 1 en position (r, i_r) (*resp.* (r, j_r)), pour r dans $[1; k]$. On a clairement $X = {}^t H_\varphi H_\psi$. On vérifie qu'on peut effectivement construire un automate $\mathcal{C}$ conjugué à $\mathcal{A}$ et $\mathcal{B}$ respectivement par ${}^t H_\varphi$ et H_ψ.

On a vu que pour construire un $\mathbb{K}$-revêtement étant donné un éclatement, on a des contraintes portant sur des sommes partielles de colonnes de la matrice de transition ; pour un $\mathbb{K}$-co-revêtement, on a des contraintes similaires portant sur des sommes partielles de lignes. Ici, on va devoir traiter sur une même matrice ces deux contraintes. La preuve consiste à montrer que ces contraintes sont cohérentes ; on montre que la construction revient à remplir des blocs de la matrice connaissant la somme de chaque ligne et de chaque colonne.

2.5 Le cas des entiers naturels

Là encore, si l'on veut que la matrice de circulation soit inversible, la seule possibilité est l'identité. La construction est donc similaire à celle présentée pour le cas booléens, excepté que l'on ne doit pas forcément obtenir des 1 ! On numérote donc de même les coefficients non nuls et on définit $K = \{(r,s) \mid s \in [1; X_{i_r,j_r}]\}$. La matrice H_φ de taille $K\times Q$ (*resp.* H_ψ de taille $K\times R$) a des coefficients 1 en position $((r,s),i_r)$ (*resp.* $((r,s),j_r)$), pour tout (r,s) dans K.

On a clairement $X = {}^tH_\varphi H_\psi$ et on vérifie qu'on peut effectivement construire un automate $\mathcal{C}$ conjugué à $\mathcal{A}$ et $\mathcal{B}$ respectivement par ${}^tH_\varphi$ et H_ψ.

2.6 Le cas des entiers relatifs

Tout entier relatif non nul n'est pas somme de 1, donc on ne peut pas appliquer la construction précédente sur $\mathbb{Z}$; par ailleurs, l'identité n'est plus la seule matrice inversible. On va donc utiliser la circulation de coefficients ± 1 pour résoudre cette difficulté. On agit de même que précédemment, sauf qu'on définit $K = \{(r,s) \mid s \in [1; |X_{i_r,j_r}|]\}$ et que D est la matrice diagonale telle que $D_{(r,r),(r,r)}$ est le signe de X_{i_r,j_r}.

On vérifie que $X = {}^tH_\varphi D H_\psi$ et on construit deux automates, $\mathcal{A}' = (I', M', T')$ conjugué à $\mathcal{A}$ par ${}^tH_\varphi$, et $\mathcal{B}' = (J', N', U')$ conjugué à $\mathcal{B}$ par H_ψ tels que $\mathcal{A}'$ et $\mathcal{B}'$ sont conjugué par D.

On peut noter que pour calculer les matrices de transition de $\mathcal{A}'$ et $\mathcal{B}'$, on calcule en fait $M'D = DN'$ et qu'on a ensuite besoin que D soit inversible.

Exemple. Les deux automates $\mathcal{A}_5 = (I, M, T)$ et $\mathcal{B}_5 = (J, N, U)$ de la figure 2.3 sont conjugués par la matrice

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} = CDL,$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \ D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \ L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Construisons les $\mathbb{Z}$-automates $\mathcal{A}' = (I', M', T')$ et $\mathcal{B}' = (J', N', U')$ respectivement co-$\mathbb{Z}$-revêtement de $\mathcal{A}$ et $\mathbb{Z}$-revêtement de $\mathcal{B}$ et égaux à circulation de coefficients près. On pose $I' = IC$, $J' = I'D$ et on a bien $J = ICDL = J'L$. De même, $U' = LU$, $T' = DU'$ et on a $T = CDLU = CU'$. On doit remplir les matrices M' et N' telles que $MC = CM'$, $M'D = DN'$ et $N'L = LN$. On pose $H = M'D = DN'$; on doit donc avoir $MCD = CH$ et $HL = DLN$:

$$\left[\begin{array}{cccccccccc}
0 & . & -a & . & a & . & . & . & 0 & . \\
 & 0 & & -a & & a & 0 & 0 & & 0 \\
-b & . & 0 & . & 0 & . & . & . & -a & . \\
b & . & 0 & . & 0 & . & . & . & -a & . \\
 & 0 & & 0 & & 0 & 0 & 0 & & -2a \\
b & . & 0 & . & 0 & . & . & . & -a & . \\
b & . & 0 & . & 0 & . & . & . & a & . \\
 & 2b & & 0 & & 0 & 0 & 0 & & 0 \\
0 & . & b & . & b & . & . & . & 0 & . \\
 & 0 & & 0 & & 0 & b & b & & 0
\end{array}\right]$$

La figure ci-dessus est une transcription des équations sur la matrice H. Pour chaque sous-matrice matérialisée par des rectangles, on connaît la somme de chaque colonne (indiquée en dessous) et de chaque ligne (indiquée à gauche). On a plusieurs possibilités pour remplir cette matrice (on a le choix pour chaque sous-matrice 2×2 ; la plus simple étant :

$$H = \begin{bmatrix}
0 & -a & a & 0 & 0 & 0 \\
-b & 0 & 0 & 0 & 0 & -a \\
b & 0 & 0 & 0 & 0 & -a \\
b & 0 & 0 & 0 & 0 & -a \\
b & 0 & 0 & 0 & 0 & a \\
0 & 0 & 0 & b & b & 0
\end{bmatrix}$$

Finalement, on obtient le diagramme suivant :

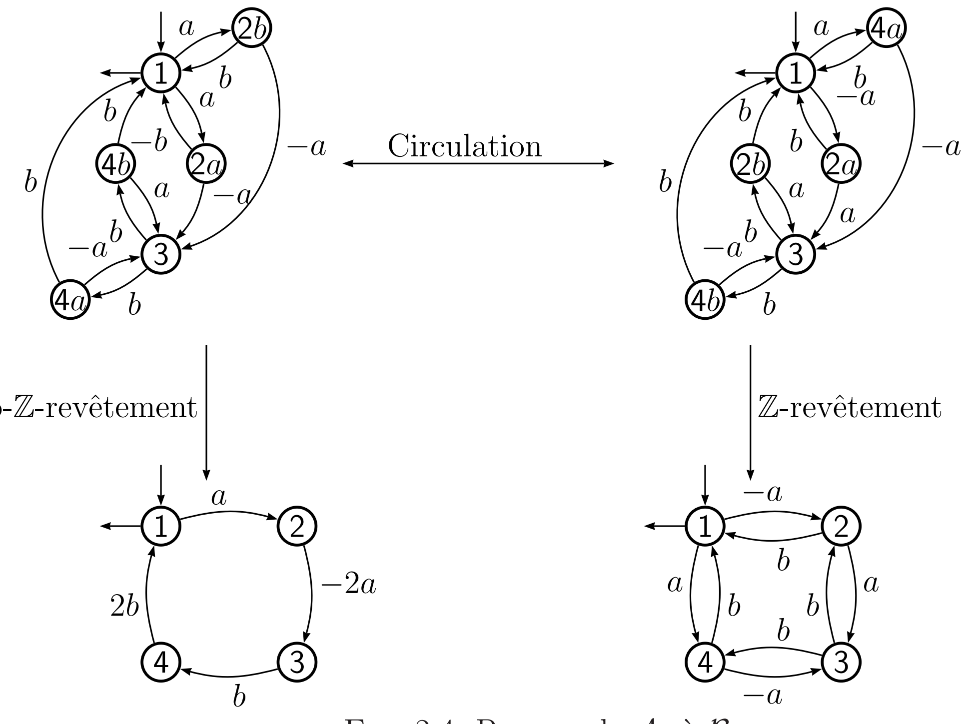


FIG. 2.4- Passage de $\mathcal{A}_5$ à $\mathcal{B}_5$.

L'état à travers lequel on fait circuler -1 est $2a$ dans les revêtements.

2.7 Le cas des corps

Dans un corps, la situation est d'une certaine manière plus simple, puisque tout élément non nul est inversible. On peut donc calculer les matrices H_φ et H_ψ exactement comme dans le cas booléen et définir la matrice diagonale D par $D_{r,r} = X_{i_r,j_r}$ pour tout r dans $[1;k]$. La suite est identique au cas des entiers relatifs.

Revenons un instant sur l'hypothèse que les automates sont émondés. Cette hypothèse assure que la matrice de conjugaison X n'a pas de ligne ou de colonne nulle, ce qui permet de construire des fonctions φ et ψ d'éclatement surjectives.

Dans le cas de $\mathbb{Z}$ ou d'un corps, comme chaque élément a un opposé, on peut s'affranchir de cette hypothèse. Plaçons-nous dans le cas de $\mathbb{Z}$. Imaginons que la ligne i de X est nulle. Il suffit de faire comme si $X_{i,1} = 2$; supposons que c'est le coefficient non nul numéro r ; on construit H_φ et H_ψ comme expliqué précédemment, et on pose $D_{(r,1),(r,1)} = 1$ et $D_{(r,2),(r,2)} = -1$.

2.8 Le cas des transducteurs

On montre maintenant que si on a $\mathcal{T} \stackrel{X}{\Longrightarrow} \mathcal{U}$, avec X matrice dont les coefficients sont des mots, il existe deux transducteurs $\mathcal{T}'$ et $\mathcal{U}'$ identiques à circulation de coefficients près tels que $\mathcal{T}'$ est un co-revêtement de $\mathcal{T}$ et $\mathcal{U}'$ un revêtement de $\mathcal{U}$.

La construction est exactement la même que pour les corps. Pour être exact, il faudrait considérer les coefficients de X comme appartenant au groupe libre, si on veut respecter la contrainte que la matrice de circulation est inversible. De plus, ceci correspond plus à la circulation usuelle des sorties dans un transducteur.

3 Synthèse

On peut combiner les deux propositions 2.1 et 2.2, afin d'obtenir un résultat dont la conjugaison est, en apparence, absente :

Théorème 2.1 $\mathbb{K} = \mathbb{Z}$ *ou un corps.*
Soit $\mathcal{A}$ et $\mathcal{B}$ deux automates (resp. deux $\mathbb{N}$-automates) émondés équivalents. Alors il existe trois automates (resp. trois $\mathbb{N}$-automates) $\mathcal{A}'$, $\mathcal{B}'$ et $\mathcal{C}$ tels que $\mathcal{A}'$ et $\mathcal{B}'$ sont des ($\mathbb{N}$-)co-quotients de $\mathcal{C}$ et $\mathcal{A}$ (resp. $\mathcal{B}$) est un quotient de $\mathcal{A}'$ (resp. $\mathcal{B}'$).
Soit $\mathcal{A}$ et $\mathcal{B}$ deux $\mathbb{K}$-automates (resp. deux transducteurs) émondés équivalents. Alors il existe quatre automates (resp. quatre transducteurs) $\mathcal{A}'$, $\mathcal{B}'$, $\mathcal{A}''$ et $\mathcal{B}''$ tels que $\mathcal{A}''$ et $\mathcal{B}''$ sont égaux à une circulation de coefficients près, $\mathcal{A}'$ et $\mathcal{B}'$ sont des ($\mathbb{K}$-)co-quotients respectifs de $\mathcal{A}''$ et $\mathcal{B}''$, et $\mathcal{A}$ et $\mathcal{B}$ sont des ($\mathbb{K}$-)quotients respectifs de $\mathcal{A}'$ et $\mathcal{B}'$.

En plus des deux propositions, ce résultat nécessite un “lemme du diamant” : si deux automates ont un même (co-)quotient, ils admettent un même (co-)revêtement.

Le résultat suivant est une conséquence quasi-directe du théorème appliqué au $\mathbb{N}$-automates.

Théorème 2.2 *Soit deux langages rationnels ayant la même fonction de croissance (i.e. le nombre de mots d'une longueur donnée est le même dans chacun des langages). Alors il existe une transduction lettre à lettre fonctionnelle qui réalise une bijection entre les deux langages.*

Exemple. Considérons les deux langages $L_1 = a(a+b)^*$ et $L_2 = (c+dc+dd)^* \setminus cc(c+d)^*$. Ces langages ont même fonction de croissance ; leurs automates minimaux sont représentés sur la figure 2.5 en bas à gauche ($\mathcal{A}$) et à droite ($\mathcal{B}$).

Pour construire le transducteur, on commence par oublier les étiquettes des transitions (ce qui revient à considérer qu'on a un alphabet unique à une lettre). On obtient ainsi les représentation suivantes :

$$I = \begin{bmatrix} 1 & 0 \end{bmatrix}, M = \begin{bmatrix} 0 & 1 \\ 0 & 2 \end{bmatrix}, T = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$J = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, N = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}, T = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Ces deux $\mathbb{N}$-automates sont équivalents, ils réalisent chacun la fonction de croissance. Calculons la réduction gauche conjointe :

$$\begin{aligned} \alpha_1 = I|J &= 10|1000 \\ \alpha_2 = \alpha_1(M|N) &= 01|0110 \\ \alpha_3 = \alpha_2(M|N) &= 02|0012 \\ \alpha_3(M|N) &= 04|0024 = 2\alpha_3 \end{aligned}$$

Ce qui donne l'automate $\mathcal{C}$ présenté en bas au centre de la figure :

$$I = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, M = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \end{bmatrix}, T = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}$$

L'automate $\mathcal{C}$ est conjugué à $\mathcal{A}$ par la matrice $X = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 2 \end{bmatrix}$. On peut donc calculer un revêtement de $\mathcal{A}$ et un co-revêtement de $\mathcal{C}$: c'est l'automate $\mathcal{A}'$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

De même, on calcule l'automate $\mathcal{B}'$ revêtement de $\mathcal{B}$ et co-revêtement de $\mathcal{C}$:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Le lemme du diamant nous permet ensuite de trouver un co-revêtement commun à $\mathcal{A}'$ et $\mathcal{B}'$. Il y a bijection entre les transitions de ce co-revêtement et celle de $\mathcal{A}$ d'une part et de

$\mathcal{B}$ d'autre part. En reportant les étiquettes, on obtient le transducteur $\mathcal{T}$ lettre-à-lettre dont les projections sur chacune de ses composantes donnent des automates non ambigus et qui réalise donc une fonction bijective.

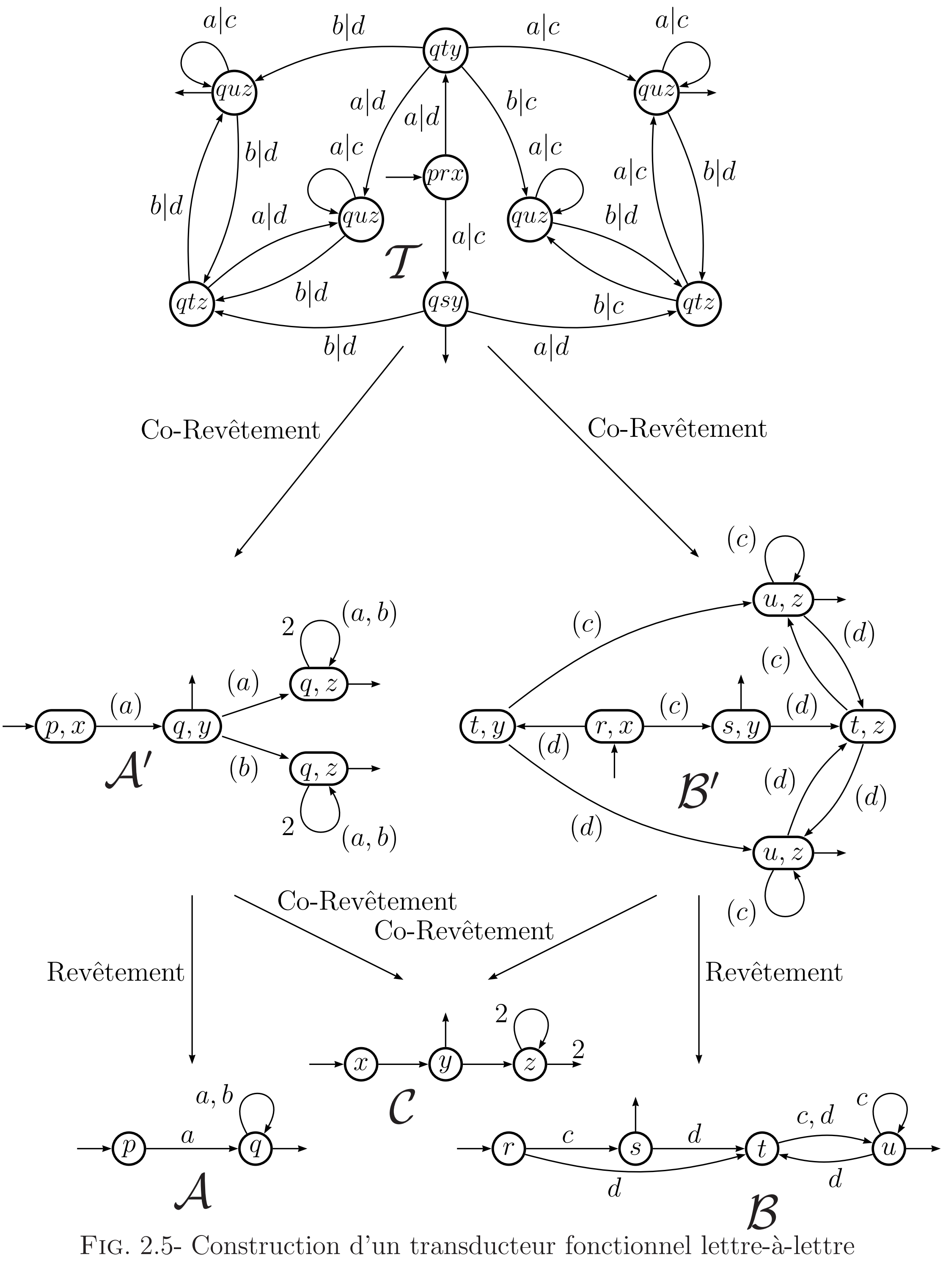


FIG. 2.5- Construction d'un transducteur fonctionnel lettre-à-lettre

4 Conjugaison en dynamique symbolique

Pour tout ce qui concerne ce paragraphe, on se référera aux ouvrages de référence sur la dynamique symbolique [3, 32]. La dynamiques symbolique manipule des mots bi-infinis. Étant donné un alphabet A, on note $A^{\mathbb{Z}}$ l'ensemble des mots bi-infinis de sur A et σ l'opération de shift :

$$\sigma : \begin{array}{ccl} A^{\mathbb{Z}} & \longrightarrow & A^{\mathbb{Z}} \\ (a_i)_{i\in\mathbb{Z}} & \longmapsto & (b_i)_{i\in\mathbb{Z}} \text{ avec } b_i = a_{i-1} \end{array}$$

Un *sous-shift* de $A^{\mathbb{Z}}$ est un sous-ensemble clos pour σ et σ^{-1} fermé pour la topologie usuelle sur $A^{\mathbb{Z}}$ (produit de la topologie discrète).

L'entropie d'un sous-shift S est la limite supérieure de $\log(S_n)/n$, où S_n est l'ensemble des facteurs de longueurs n de S.

Un sous-shift est *sofique* s'il correspond à l'ensemble des étiquettes d'un système de transition fini (un automate sans état initial ou terminal). On peut alors représenter ce sous-shift avec un système déterministe (soluble à droite) ou co-déterministe (soluble à gauche) en appliquant des algorithmes de déterminisation par la méthode des sous-ensemble.

Il est *de type fini* s'il est caractérisé par un ensemble fini de facteurs interdits. En particulier, l'ensemble des chemins d'un système de transition fini est un sous-shift de type fini.

Soit K (*resp.* L) un sous-shift de $A^{\mathbb{Z}}$ (*resp.* $B^{\mathbb{Z}}$). $\Phi : K \to L$ est une application par bloc s'il existe m et a dans $\mathbb{N}$ et $\varphi : A^{m+1+a} \to B$ tel que pour tout u dans L, $v = \Phi(u)$ si et seulement si pour tout i, $v_i = \varphi(u_{i-m}...u_{i+a})$. Une telle application est *finite-to-one* si chaque élément de L a un nombre borné de pré-images.

Une application *finite-to-one* remarquable est celle qui envoie un sous-shift de type fini (correspondant à l'ensemble des chemins d'un système de transition) sur le sous-shift sofique obtenu en étiquetant ce système de sorte à obtenir un système déterministe.

On dit que deux sous-shift sofiques sont finiment équivalents s'il exite un sous-shift de type fini dont ils sont les images par des applications par bloc *finite-to-one*.

Un théorème important de la dynamique symbolique est le théorème de l'équivalence finie qui dit que deux sous-shift sofiques irréductibles (*i.e.* représentés par un système de transition fortement connexe) sont finiment équivalents si et seulement si ils ont la même entropie.

La preuve de ce théorème consiste d'abord à se ramener aux sous-shifts de type finis, car les applications *finite-to-one* préservent l'entropie et ensuite à utiliser le lemme de Furstenberg qui établit que deux sous-shifts de type finis de même entropie vérifie l'équation suivante : si X et Y sont les matrices des graphes (fortement connexes) correspondants aux sous-shifts, il existe une matrice positive non nulle F telle que $XF = FY$.

On voit réapparaître notre équation de conjugaison. Le cœur de la preuve consiste alors à montrer que cette équation implique l'équivalence finie des deux sous-shifts, c'est-à-dire l'existence d'un sous-shifts dont ils sont l'image par des applications *finite-to-one*.

On va voir que ce résultat a un analogue en théorie des automates ; les applications *finite-to-one* étant remplacées par des revêtements et co-revêtements. On pourrait pointer des correspondances encore plus précises, puisqu'on peut faire en sorte que l'une des applications finite-to-one préserve le déterminisme (comme les revêtements) et l'autre le co-déterminisme (comme les co-revêtements).

Chapitre 3

Dérivations d'expressions rationnelles

Une des caractérisations des séries rationnelles porte sur leurs quotients (ou résiduels). Le quotient (gauche) d'une série s par un mot u est la série $u^{-1}s = \sum_{v \in A^*} \langle s, uv \rangle v$. Une série est rationnelle si et seulement si l'ensemble de ses quotients appartient à un semi-module finiment engendré.

De nombreux problèmes sur les automates se ramènent à décrire le plus exactement possible ce semi-module. Trouver un automate de taille minimale revient à trouver un ensemble générateur minimal, savoir si la série est séquentielle est la même chose que savoir si ce semi-module est un cône finiment engendré, *etc.*

En effet, tout ensemble générateur de ce semi-module correspond à un automate, et inversement, tout automate (I, μ, T) induit un morphisme : $\Phi : \alpha \to \sum_w (\alpha\mu(w)T)w$ dont l'image est un module finiment engendré (par les $\Phi(e_i)$) contenant les quotients de la série s réalisée par l'automate ($w^{-1}s = \Phi(I\mu(w))$).

La dérivation des expressions rationnelle est une méthode qui, à partir d'une expression rationnelle décrivant une série, permet de construire un ensemble fini d'expressions représentant les générateurs d'un module contenant les quotients de la série.

J. Brzozowski [8] eut le premier l'idée de définir les dérivées d'une expression rationnelle permettant de représenter *tous* les quotients du langages ; il en déduit ainsi la construction d'un automate déterministe. Cet automate est non minimal d'ailleurs car il se peut qu'on obtienne plusieurs dérivées correspondant au même quotient.

Cette méthode a deux inconvénients. D'abord, elle nécessite d'appliquer aux expressions les équivalences induites par l'associativité des opérations d'une part et la commutation et l'idempotence de l'addition d'autre part, ce qui alourdit considérablement les calculs (on appelle l'ensemble de ces équivalences ACI). De plus, elle n'est pas généralisable aux séries, puisque les quotients d'une série rationnelle ne sont généralement pas en nombre fini.

V. Antimirov [1] a proposé une modification de la dérivation, qu'il a appelée dérivation partielle et qui permet de calculer un nombre moins important d'expressions (donc d'obtenir un automate plus petit mais non déterministe). Le résultat d'une dérivation n'est plus alors une expression mais un ensemble d'expressions. De plus, cette méthode permet de ne plus recourir aux équivalences ACI. C'est cet algorithme que nous avons généralisé.

1 Dérivation des expressions avec multiplicités

On se place donc dans le cadre des expressions rationnelles avec multiplicité. La dérivée d'une expression par une lettre a est un polynôme d'expressions défini de la façon suivante :

$$\frac{\partial}{\partial a}\mathsf{0} = \frac{\partial}{\partial a}\mathsf{1} = \mathsf{0}, \qquad \frac{\partial}{\partial a} b = \begin{cases} \mathsf{1} & \text{if} \quad b = a \\ \mathsf{0} & \text{sinon} \end{cases}$$

$$\frac{\partial}{\partial a}(k\,\mathsf{E}) = k\,\frac{\partial}{\partial a}\mathsf{E} \qquad \frac{\partial}{\partial a}(\mathsf{E}\,k) = \left(\frac{\partial}{\partial a}\mathsf{E}\,k\right)$$

$$\frac{\partial}{\partial a}(\mathsf{E}+\mathsf{F}) = \frac{\partial}{\partial a}\mathsf{E} \oplus \frac{\partial}{\partial a}\mathsf{F} \qquad \frac{\partial}{\partial a}(\mathsf{E}\cdot\mathsf{F}) = \left(\left[\frac{\partial}{\partial a}\mathsf{E}\right]\cdot\mathsf{F}\right) \oplus \mathsf{c}(\mathsf{E})\,\frac{\partial}{\partial a}\mathsf{F}$$

$$\frac{\partial}{\partial a}(\mathsf{E}^*) = \mathsf{c}(\mathsf{E})^*\left(\left[\frac{\partial}{\partial a}\mathsf{E}\right]\cdot(\mathsf{E}^*)\right)$$

L'idée, on le voit, est de “manger” dans l'expression la première lettre a pouvant être le début d'un mot du langage. On peut traverser le facteur gauche E d'une expression si celui-ci représente une série dont le support contient le mot vide ; il faut alors tenir compte du coefficient du mot vide de E : $\mathsf{c}(\mathsf{E})$. Ce coefficient est facilement calculable à partir de l'expression.

La dérivation par rapport à un mot est définie en itérant la dérivation par rapport aux lettres du mot. On montre que si E représente la série $|\mathsf{E}| = s$ et que $\frac{\partial}{\partial w}\mathsf{E} = \bigoplus k_i\mathsf{E}_i$, la série $w^{-1}s$ est égale à $\bigoplus k_i|\mathsf{E}_i|$.

D'autre part, les expressions qui apparaissent dans l'ensemble des dérivations d'une expression E, qu'on appelle *termes dérivés* de E sont en nombre fini (borné par la longueur littérale de E).

On peut ainsi calculer un ensemble fini d'expressions rationnelles K_i, clos par dérivation contenant tous les termes dérivés de E. Le fait que tous les K_i ne sont pas des termes dérivés peut se produire dans certains semi-anneaux ; c'est un problème que je ne développerai pas ici (*cf.* [43]).

On construit l'automate $\mathcal{A}_{\mathsf{E}} = (I, \mu, T)$ des termes dérivés de E de la façon suivante :
– L'ensemble d'états est $Q = \{\mathsf{E}, \mathsf{K}_1, ...\mathsf{K}_m\}$;
– $I_{\mathsf{E}} = 1$, sinon $I_{\mathsf{K}_i} = 0$;
– μ est tel que $\frac{\partial}{\partial a}\mathsf{F} = \bigoplus \mu(a)_{\mathsf{F},\mathsf{K}_i}\mathsf{K}_i$;
– T_{F} est le coefficient du mot vide dans $|\mathsf{F}|$.

Théorème 3.1 *[39, 43] L'automate des termes dérivés d'une $\mathbb{K}$-expression rationnelle de longueur littérale n a au plus $n+1$ états. Plus précisément, c'est un $\mathbb{K}$-quotient de l'automate standard (ou de Glushkov) de cette expression.*

La définition de l'automate standard d'une expression avec multiplicité a été donnée par P. Caron et M. Flouret [9].

Exemple. Soit $\mathsf{E}_5 = (2ab + (3b) \cdot (4ab)^*)^*$. Calculons les termes dérivés :

$$\frac{\partial}{\partial a}\mathsf{E}_5 = 2b \cdot \mathsf{E}_5 = 2\mathsf{K}_1 \qquad \frac{\partial}{\partial b}\mathsf{E}_5 = 3(4ab)^* \cdot \mathsf{E}_5 = 3\mathsf{K}_2$$
$$\frac{\partial}{\partial a}\mathsf{K}_1 = 0 \qquad \frac{\partial}{\partial b}\mathsf{K}_1 = \mathsf{E}_5$$
$$\frac{\partial}{\partial a}\mathsf{K}_2 = 4(b \cdot (4ab)^*) \cdot \mathsf{E}_5 = 4\mathsf{K}_3 \qquad \frac{\partial}{\partial b}\mathsf{K}_2 = 0$$
$$\frac{\partial}{\partial a}\mathsf{K}_3 = 0 \qquad \frac{\partial}{\partial b}\mathsf{K}_3 = (4ab)^* \cdot \mathsf{E}_5 = \mathsf{K}_2$$

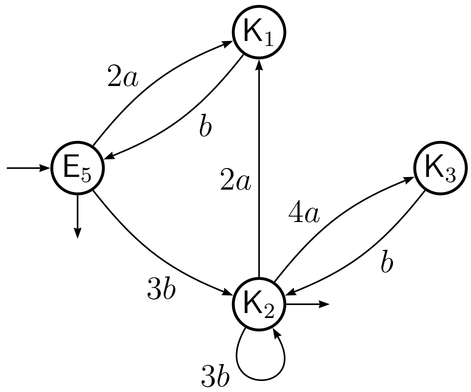


Fig. 3.1- L'automate des termes dérivés de E_5

2 Comment les expressions rationnelles codent les automates

Revenons aux expressions rationnelles sans multiplicité. Le calcul d'une expression rationnelle à partir d'un automate peut donner un résultat dont la taille est exponentielle par rapport à celle de l'automate.

Nous avons remarqué que sur une telle expression, l'algorithme de dérivation donne un automate dont la taille est comparable à celle de l'automate de départ, ce qui est déjà remarquable, mais que, de plus, cet automate est bien souvent celui à partir duquel l'expression a été calculée. Nous avons donc cherché à expliquer ce phénomène, et à modifier la définition de la dérivation donnée par V. Antimirov [1], afin de trouver un algorithme qui soit le plus proche possible d'un inverse du calcul de l'expression rationnelle.

L'algorithme que nous utilisons pour calculer une expression rationnelle à partir d'un automate, est indifféremment l'algorithme de McNaughton-Yamada [46], ou celui qui consiste à éliminer les états en tenant à jour les étiquettes des transitions, ou encore celui qui revient à résoudre un système linéaire. Pour plus d'explication sur ces algorithmes, on se référera à [52], par exemple. Le déroulement de chacun de ces algorithmes suppose un ordre sur les états, et deux ordres différents donnent une expression différente. Par ailleurs, pour un ordre fixé, ces trois algorithmes donnent des expressions semblables (*cf.* [52]).

Il apparaît en premier lieu que l'automate des termes dérivés tel que nous l'avons défini ci-dessus n'a qu'un état initial, ce qui restreint fortement les chances d'atteindre notre objectif. C'est donc pour cette raison que nous avons défini sur les expressions une opération qu'on peut interpréter comme une dérivation par rapport au mot vide et dont le rôle est de “casser” les expressions qui sont des sommes en un ensemble d'expressions :

$$D(\mathsf{0}) = \mathsf{0}, \quad D(\mathsf{1}) = \mathsf{1}, \quad D(a) = a,$$
$$D((\mathsf{E}+\mathsf{F})) = D(\mathsf{E}) \oplus D(\mathsf{F}), \quad D((\mathsf{E}\cdot\mathsf{F})) = ([D(\mathsf{E})]\cdot\mathsf{F}),$$
$$D((\mathsf{E}^*)) = (\mathsf{E}^*)$$

On définit ainsi une nouvelle dérivation, dite cassante : $\frac{\partial_c}{\partial a}\mathsf{E} = D(\frac{\partial}{\partial a}\mathsf{E})$.

Si E est une expression, on calcule l'ensemble des termes dérivés initiaux $D(\mathsf{E})$. Les autres termes dérivés sont ensuite obtenus à partir des termes dérivés initiaux en appliquant la dérivation cassante. Un automate est construit de la même manière que l'automate des termes dérivés ; les états initiaux en sont les termes dérivés initiaux.

Cet automate a la propriété remarquable suivante :

Proposition 3.1 *Si $\mathcal{A}$ est un automate co-déterministe et E est une expression rationnelle calculée à partir de $\mathcal{A}$, l'automate des termes dérivés cassants est co-déterministe.*

La propriété de co-déterminisme survit donc à la transformation en expression rationnelle.

Notons immédiatement que l'on peut définir une dérivation à droite des mots qui permet de retrouver un automate déterministe à partir d'une expression calculée sur un automate déterministe, ce qui permet d'éviter une étape, éventuellement exponentielle, de déterminisation.

Exemple. Considérons l'automate $\mathcal{A}_6$ ci-dessous :

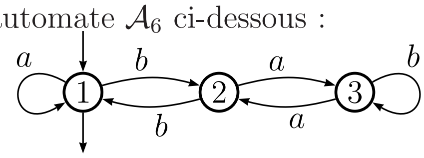


Fig. 3.2- Diviseur par 3 en base 2

Le calcul d'une expression rationnelle par l'élimination des états donne sur cet automate :

$$\mathsf{E}_6 = a^* + a^*b(ba^*b)^*ba^* + a^*b(ba^*b)^*a(b + a(ba^*b)^*a)^*a(ba^*b)^*ba^*$$

On pose $\mathsf{F} = (ba^*b)^*ba^*$ et $\mathsf{G} = (b + a(ba^*b)^*a)^*$; les termes dérivés initiaux de E_6 sont :

$$\mathsf{E}_6^{(1)} = a^*, \quad \mathsf{E}_6^{(2)} = a^*b(ba^*b)^*ba^* = a^*b\mathsf{F}$$
$$\mathsf{E}_6^{(3)} = a^*b(ba^*b)^*a(b + a(ba^*b)^*a)^*a(ba^*b)^*ba^* = a^*b(ba^*b)^*a\mathsf{G}a\mathsf{F}$$

Calculons les termes dérivés :

$$\frac{\partial}{\partial a}\mathsf{E}_6^{(1)} = a^* = \mathsf{E}_6^{(1)} \qquad \frac{\partial}{\partial b}\mathsf{E}_6^{(1)} = 0$$
$$\frac{\partial}{\partial a}\mathsf{E}_6^{(2)} = a^*b\mathsf{F} = \mathsf{E}_6^{(2)} \qquad \frac{\partial}{\partial b}\mathsf{E}_6^{(2)} = \mathsf{F} = \mathsf{K}_1$$
$$\frac{\partial}{\partial a}\mathsf{K}_1 = 0 \qquad \frac{\partial}{\partial b}\mathsf{K}_1 = a^*b\mathsf{F} \cup a^* = \mathsf{E}_6^{(2)} \cup \mathsf{E}_6^{(1)}$$
$$\frac{\partial}{\partial a}\mathsf{E}_6^{(3)} = a^*b(ba^*b)^*a\mathsf{G}a\mathsf{F} = \mathsf{E}_6^{(3)} \qquad \frac{\partial}{\partial b}\mathsf{E}_6^{(3)} = (ba^*b)^*a\mathsf{G}a\mathsf{F} = \mathsf{K}_2$$
$$\frac{\partial}{\partial a}\mathsf{K}_2 = \mathsf{G}a\mathsf{F} = \mathsf{K}_3 \qquad \frac{\partial}{\partial b}\mathsf{K}_2 = a^*b(ba^*b)^*a\mathsf{G}a\mathsf{F} = \mathsf{E}_6^{(3)}$$
$$\frac{\partial}{\partial a}\mathsf{K}_3 = (ba^*b)^*a\mathsf{G}a\mathsf{F} \cup \mathsf{F} = \mathsf{K}_2 \cup \mathsf{K}_1 \qquad \frac{\partial}{\partial b}\mathsf{K}_3 = \mathsf{G}a\mathsf{F} = \mathsf{K}_3$$

L'automate des termes dérivés qu'on obtient est :

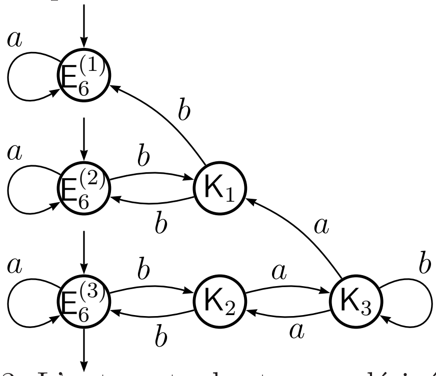


FIG. 3.3- L'automate des termes dérivés de E_6

Remarquons tout de suite que l'expression E_6 a une longueur littérale de 26, et que l'automate des termes dérivés n'a que 6 états, alors que l'automate standard en aurait 27. Cet automate n'est pas exactement l'automate de départ, même s'il y ressemble. En effet, il suffit de calculer le co-quotient minimal, c'est-à-dire fusionner les états qui ont les mêmes entrées pour retrouver l'automate des diviseurs par 3.

La réussite de l'opération est ici garantie par la proposition 3.1, puisque l'automate des diviseurs est co-déterministe minimal.

Malheureusement, cette procédure ne donne pas toujours le résultat attendu ; pour différents contre-exemples ainsi que des pistes pour résoudre les différents problèmes, on se référera à [44].

Chapitre 4

Réversibilité et hauteur d'étoile

Ce chapitre rapporte des travaux effectués durant ma thèse.

1 Hauteur d'étoile et enlacement

La hauteur d'étoile d'une expression rationnelle E est le nombre maximal d'étoiles enchâssées dans cette expression :

$$\begin{gathered}\mathsf{h}(0) = \mathsf{h}(1) = \mathsf{h}(a) = 0 \\ \mathsf{h}(\mathsf{E} + \mathsf{F}) = \mathsf{h}(\mathsf{E} \cdot \mathsf{F}) = \mathsf{max}(\mathsf{h}(\mathsf{E}), \mathsf{h}(\mathsf{F})) \\ \mathsf{h}(\mathsf{E}^*) = \mathsf{h}(\mathsf{E}) + 1\end{gathered}$$

Le problème de la hauteur d'étoile consiste, étant donné un langage rationnel, à trouver une expression de hauteur d'étoile minimum pour ce langage. Il s'agit d'un vieux problème soulevé par L.C. Eggan [15] et dont la réponse fut apportée par K. Hashiguchi [21] et, de manière plus satisfaisante, plus récemment par D. Kirsten [24] (il donne un algorithme doublement exponentiel).

L.C. Eggan [15] avait montré que le problème de la hauteur d'étoile peut être exprimé sur les automates qui reconnaissent le langage. En effet, plus un automate a de circuits imbriqués les uns dans les autres, plus la hauteur d'étoile des expressions correspondantes sera importante, et vice-versa.

La mesure du nombre de circuits imbriqués est appelé *enlacement* :

- l'enlacement d'un automate acyclique est nul ;
- l'enlacement d'un automate non fortement connexe est le maximum des enlacements de ses composantes fortement connexes ;
- l'enlacement d'une composante fortement connexe (avec au moins une transition) est égale à l'automate d'enlacement minimum obtenu par effacement de l'un de ses états, plus 1.

Nous avons donné une nouvelle présentation de ce résultat en [41].

R. McNaughton [45] a montré que si un morphisme d'automate est surjectif sur les chemins, il préserve la hauteur d'étoile. En d'autres termes, si $\mathcal{A}$ et $\mathcal{B}$ sont deux automates et φ un morphisme de $\mathcal{A}$ sur $\mathcal{B}$ tel que tout chemin de $\mathcal{B}$ peut se relever dans $\mathcal{A}$, alors $\mathcal{B}$ a un enlacement inférieur ou égal à celui de $\mathcal{B}$. On verra plus loin comment ce résultat peut être exploité pour résoudre certains des problèmes concernant la hauteur d'étoile.

2 Automate universel

L'automate universel $\mathcal{U}_l$ d'un langage rationnel est formé à partir des *factorisations* du langage, introduite en 1971 par J.H. Conway [14]. Une factorisation de L est un couple de langages (G, D) tel que $G.D \subseteq L$ et qui est maximal (pour l'inclusion) parmi ceux qui respectent cette propriété.

Le nombre de factorisations d'un langage rationnel est fini et on peut donc construire un automate dont elles sont les états :
- (G, D) est initial si le mot vide est dans G ;
- (G, D) est final si le mot vide est dans D ;
- il y a une transition étiquetée par a de (G, D) à (G', D') si $G.a.D'$ est inclus dans L.

L'automate $\mathcal{U}_L$ reconnaît évidemment L.

On peut construire $\mathcal{U}_L$ à partir du *monoïde syntaxique* de L ; on peut toutefois s'affranchir de calculer ce monoïde : j'ai donné une construction de cet automate à partir de l'automate minimal $\mathcal{A}_L = (\{i\}, \mu, T)$ complet du langage [35, 33] :
- calculer l'ensemble des vecteurs $\mu(w)T$;
- calculer la clôture de cet ensemble par le ET (produit d'Hadamard) entre vecteurs ;
- l'ensemble obtenu donne les états de l'automate universel :
  - α est initial si $\alpha_i = 1$;
  - α est final si $\alpha \subseteq T$;
  - (α, a, β) est une transition si $\alpha\mu(a) \subseteq \beta$.

L'automate $\mathcal{U}_L$ est universel en cela que pour tout automate $\mathcal{A}$ qui reconnaît le langage L, il existe un morphisme de $\mathcal{A}$ dans $\mathcal{U}_L$. Cet automate est même le plus petit automate qui reconnaît L et qui a cette propriété. Cette propriété permet par exemple d'affirmer que l'automate universel contient le plus petit automate qui reconnaît le langage (*cf.* [2]).

Ceci suggère une méthode générale pour attaquer certains problèmes. Soit L un langage rationnel dont on se demande s'il existe un automate qui a une certaine propriété $\mathfrak{P}$ qui reconnaît L. Si un tel automate $\mathcal{A}$ existe, il existe un morphisme de $\mathcal{A}$ dans $\mathcal{U}_L$, et si ce morphisme est bien choisi, avec un peu de travail, on peut montrer que l'image de $\mathcal{A}$ a aussi la propriété $\mathfrak{P}$ ou des traces de cette propriété.

3 Langages réversibles

J'ai appliqué cette méthode aux langages réversibles.

Un automate est réversible si, abstraction faite de ses états initiaux et terminaux, il est à la fois déterministe et co-déterministe.

Si en plus cet automate est complet, c'est un automate *à groupe* (car son monoïde syntaxique est dans ce cas un groupe).

Un langage est réversible (*resp.* à groupe) s'il peut être reconnu par un automate réversible (*resp.* à groupe).

Décider si un langage rationnel est un langage à groupe n'est pas difficile, car, dans ce cas, son automate minimal est un automate à groupe. Pour les langages réversibles, les choses ne sont pas aussi simples. J.-É. Pin [49] a donné une caractérisation de ces langages basée sur leurs monoïdes syntaxiques, ce qui permet la décidabilité de l'appartenance à cette classe.

Théorème 4.1 *[35] Si L est un langage réversible, l'automate universel de L a des composantes fortement connexes réversibles et contient un sous-automate* quasi-réversible *qui reconnaît L.*

Un automate est quasi-réversible si la non réversibilité ne porte que sur des transitions qui n'appartiennent à aucune composante fortement connexe. À partir d'un tel automate, on peut facilement construire un automate réversible (ce que ne permet pas la caractérisation par monoïde syntaxique).

L'idée de la preuve est la suivante : on sait qu'il existe un automate réversible qui reconnaît le langage ; on le découpe en une union d'automates réversibles dont les composantes fortement connexes sont successives (ce qui revient à isoler les différents chemins d'un graphe acyclique) ; on montre qu'on peut envoyer chacun de ces automates sur un sous-automate réversible de l'automate universel et qu'on peut faire en sorte que cette image recouvre toutes les composantes fortement connexes qu'elle rencontre. La superposition des différentes images donne un automate quasi-réversible.

Une stratégie similaire permet de répondre au problème de la hauteur d'étoile dans le cas des langages réversibles.

Théorème 4.2 *[40, 33] Si L est un langage réversible, son automate universel contient un sous-automate d'enlacement minimal qui reconnaît L. De plus, il existe un tel sous-automate quasi-réversible.*

Ce résultat, bien que partiel, apporte par rapport aux autres résultats connus sur la hauteur d'étoile, des informations structurelles sur la forme de l'automate d'enlacement minimum (et donc de l'expression rationnelle de hauteur d'étoile minimale).

Le cas de la hauteur d'étoile des langages à groupe avait été résolu par R. McNaughton [45]. Sa preuve peut être ré-exprimée dans le cadre de l'automate universel [41]. Ainsi, l'automate universel d'un langage à groupe a des composantes fortement connexes à groupe ; une union finie d'entre elles forment un automate d'enlacement minimum qui reconnaît le langage.

Exemple. On considère l'automate minimal $\mathcal{A}_3$ ci-dessous :

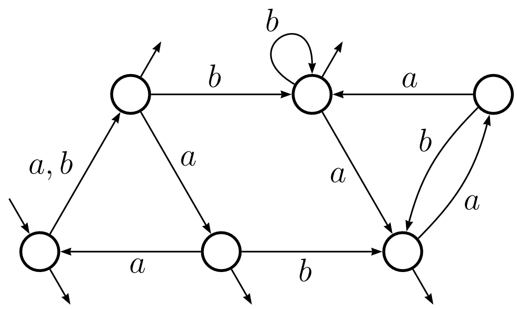


Fig. 4.1- L'automate minimal du langage L_3

L'automate universel est présensenté dans une version simplifiée ; le nombre de transitions rendrait la figure difficilement lisible. Les flèches en pointillés représentent des transitions spontanées. Toute transition ou flèche initiale ou finale qui peut être obtenue par clôture transitive (avant et arrière) selon ces transitions spontanées est une transition de l'automate universel. Par exemple, il y a une transition de $1:2$ à $5:1$ étiquetée par a dans l'automate universel car sur le dessin, on a les ε-transitions entre les couples d'états $(1{:}2, 2{:}1)$, $(2{:}2, 3)$ et $(3, 5{:}1)$ ainsi qu'une transition $(2{:}1, a, 2{:}2)$.

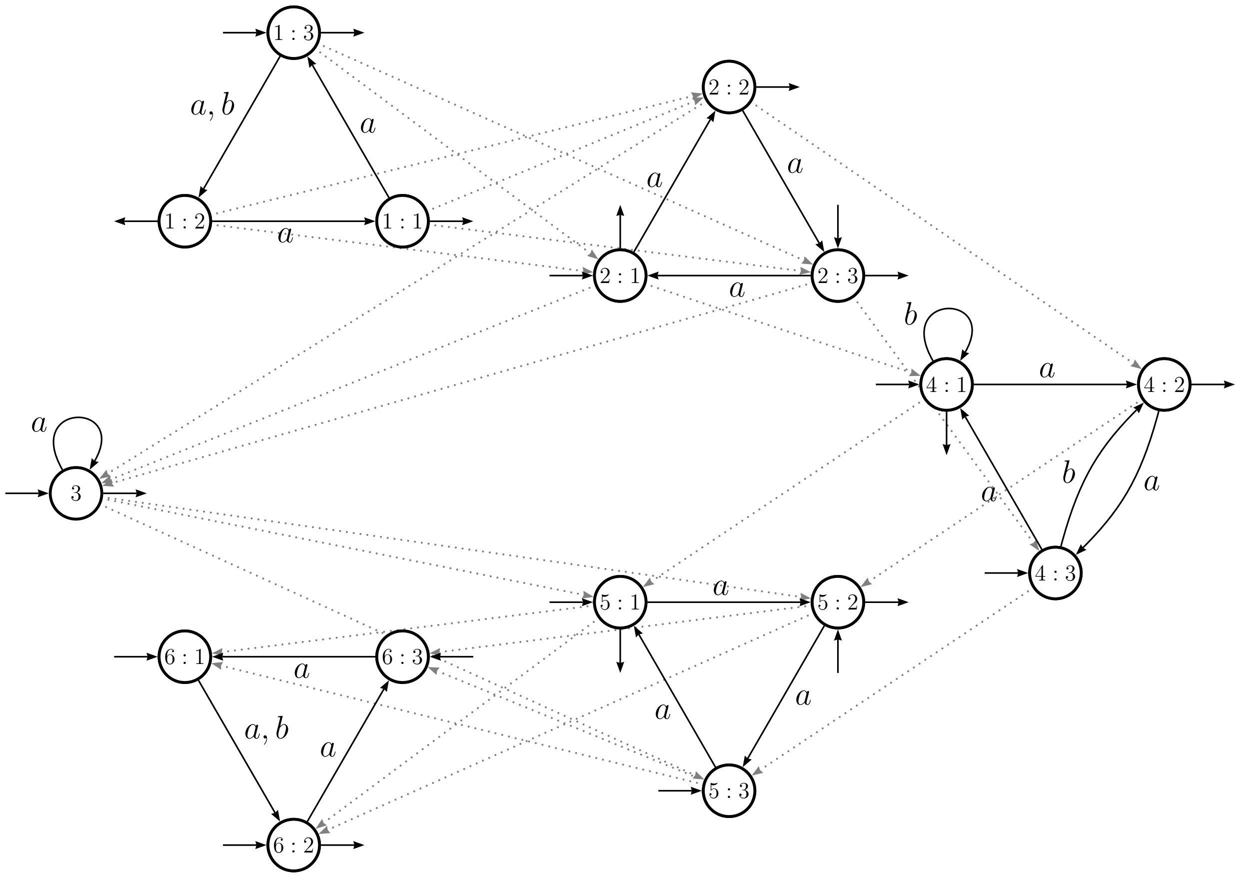


Fig. 4.2- L'automate universel de L_3.

Dans cet automate universel, on trouve le sous-automate quasi-réversible maximal suivant :

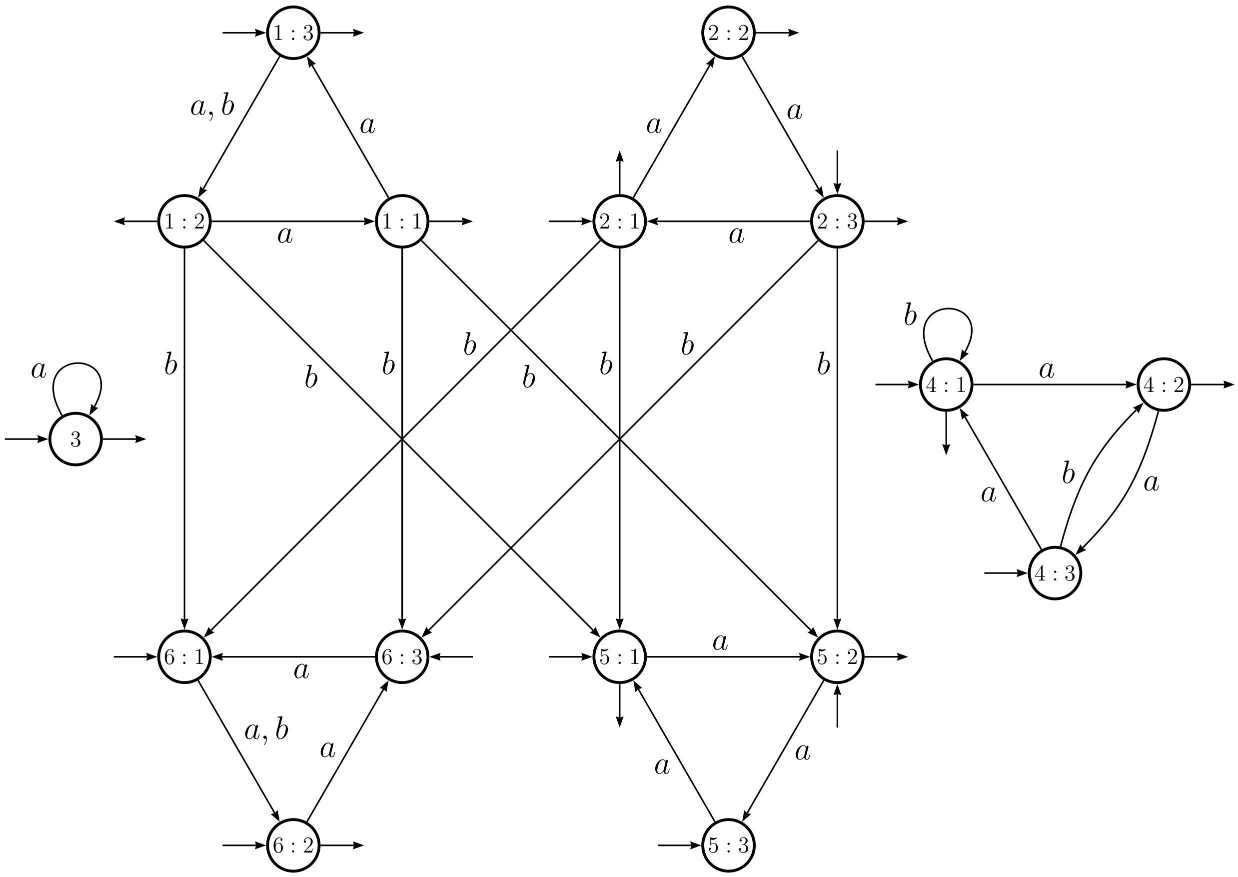


FIG. 4.3- Un automate quasi-réversible pour L_3.

Cet automate reconnaît tout le langage ; il s'agit donc d'un langage réversible. On peut se convaincre facilement que la composante fortement connexe 4 est nécessaire pour reconnaître certains mots du langages (*bbbb* par exemple) ; son enlacement est égal à 2, qui est donc la hauteur d'étoile du langage.

Chapitre 5

Automates max-plus

1 Définition

Les semi-anneaux max-plus ou min-plus sont une famille de semi-anneaux idempotents. Si $\mathbb{K}$ est un sous-monoïde additif de $\mathbb{R}$, on définit le semi-anneau max-plus $\mathfrak{K} = (\mathbb{K} \cup \{-\infty\}, \mathsf{max}, +)$ de la façon suivante :

$$\begin{aligned} \forall x, y \in \mathfrak{K}, \quad x \oplus y &= \mathsf{max}(x, y), & x \otimes y &= x + y, \\ x \oplus -\infty &= -\infty \oplus x = x, & x \otimes -\infty &= -\infty \otimes x = -\infty. \end{aligned}$$

Le *zéro* du semi-anneau est donc $-\infty$, alors que 0 en est l'unité. On considérera en particulier les semi-anneaux $\mathfrak{N}$, $\mathfrak{Z}$ et $\mathfrak{R}$ de support respectifs $\mathbb{N} \cup \{-\infty\}$, $\mathbb{Z} \cup \{-\infty\}$ et $\mathbb{R} \cup \{-\infty\}$.

On peut de même définir un semi-anneau min-plus $(\mathbb{K} \cup \{\infty\}, \mathsf{min}, +)$ de façon similaire. Pour tout semi-anneau max-plus, le passage à l'opposé est un isomorphisme vers un semi-anneau min-plus. Nous donnerons les énoncés pour le semi-anneau max-plus $\mathfrak{R} = (\mathbb{R} \cup \{-\infty\}, \mathsf{max}, +)$, ils sont en fait valable pour n'importe quel semi-anneau max-plus ou min-plus.

Il y a de nombreuses raisons d'étudier les automates à multiplicité dans ces semi-anneaux. Ce sont des objets qui apparaissent naturellement lors de problèmes d'ordonnancement ou de files d'attente. Une théorie d'algèbre linéaire assez élégante a d'ailleurs été développée pour ces semi-anneaux. C'est aussi grâce à eux que des problèmes tels que la hauteur d'étoile ont pu être résolus. Ils présentent un intérêt propre en théorie des automates car ils se situent pour bien des problèmes à la limite du décidable et de l'indécidable. D. Krob [28] a montré que l'équivalence de deux $\mathfrak{Z}$-automates est indécidable. Je me suis intéressé au problème de la déterminisation de ces automates, problème qui reste aujourd'hui encore largement ouvert.

Mon intérêt au départ pour ces semi-anneaux vient du fait qu'ils sont les semi-anneaux idempotents les plus simples (hormis le semi-anneau de Boole, bien sûr). Un certain nombre de problèmes que l'on peut se poser sur les transducteurs non fonctionnels doivent donc d'abord être résolus sur les automates max-plus. En effet, le comportement de tout $\mathfrak{N}$-automate peut être simulé par un transducteur tel que le poids k d'une transition est remplacée par la sortie de tout les mots a^r pour $r \leqslant k$.

2 Décomposition en automates non ambigus

Les séries rationnelles à coefficients dans $\mathfrak{R}$ peuvent être classifiée des plus simples (les polynômes) aux plus compliquées (qui ne sont rien d'autre que... rationnelles). Le diagramme suivant résume le paysage ainsi obtenu :

$$\mathsf{Pol} \subset \mathsf{Lim} \subset \mathsf{Seq} \begin{matrix} \subset \\ \subset \end{matrix} \begin{matrix} \mathsf{FSeq} \\ \overline{\mathsf{Amb}} = \mathsf{1val} \end{matrix} \begin{matrix} \subset \\ \subset \end{matrix} \mathsf{F}\overline{\mathsf{Amb}} = k\mathsf{val} \subset \mathsf{Rat}$$

Pol : Famille très simple ; ce sont les séries à support fini, c'est-à-dire les polynômes. L'appartenance à cette classe est décidable quelle que soit la représentation rationnelle (expression, automate) que l'on se donne.

Lim : Les séries rationnelles bornées max-plus prennent en fait un nombre fini de valeurs ; de plus, l'ensemble des mots associés à une valeur donnée est rationnel ; cette classe est décidable parmi les rationnels ; ce résultat, compliqué, est dû à K. Hashiguchi [20] ; une preuve plus simple en a été donnée par H. Leung [31].

Seq : Les séries séquentielles, c'est-à-dire réalisables par un automate déterministe ; on ne sait pas à l'heure actuelle décider l'appartenance à cette classe.

FSeq : Somme finie de séries séquentielles.

$\overline{\mathsf{Amb}} = \mathsf{1val}$: Les séries réalisables par automate non ambigu et les séries réalisables par automate 1-valué (pour chaque mot accepté, tous les chemins réussis ont la même valeur) sont en fait les mêmes. Ce résultat est en fait le même que celui qui établit que les transduction fonctionnelles sont non ambiguës ; on peut le voir comme une instance du théorème de *cross-section* rationnelle. Noter que cette famille et la précédente sont incomparables.

$\mathsf{F}\overline{\mathsf{Amb}}$ =kval : Les séries qui sont sommes finies de séries non ambiguës sont les mêmes que les séries réalisables par un automate tel que le nombre de valeurs prises par les chemins réussis étiquetés par un mot donné est borné. Ce résultat est dû à A. Weber [54].

Je présente ici une construction qui permet d'une part de transformer un automate 1-valué en un automate non ambigu, et d'autre part de transformer un automate k-ambigu, c'est-à-dire qui a au plus k chemins réussis pour tout mot accepté, en une union finie d'automates non ambigus.

Soit $\mathcal{A} = (I, \mu, T)$ un automate max-plus d'ensemble d'états Q. Soit $\mathcal{D} = (\{I\}, \delta, U)$ le déterminisé de l'automate sous-jacent de $\mathcal{A}$ d'ensemble d'états $R \subseteq \mathcal{P}(Q)$. Le revêtement de Schützenberger de $\mathcal{A} = (I, \mu, T)$ est l'automate $\mathcal{S} = (J, \nu, V)$ d'ensemble d'états $S = \{(p, P) \in Q \times R \mid p \in P\}$ défini par :

$$\begin{aligned} S_{(p,P)} &= \begin{cases} I_p & \text{si } P = I, \\ 0 & \text{sinon;} \end{cases} \\ \nu(a)_{(p,P),(q,P')} &= \begin{cases} \mu(a)_{p,q} & \text{si } \delta(a)_{P,P'} = 1, \\ 0 & \text{si } \delta(a)_{P,P'} = 0; \end{cases} \\ V_{(p,P)} &= T_p. \end{aligned}$$

Il est immédiat que les chemins réussis de $\mathcal{S}$ sont en bijection avec ceux de $\mathcal{A}$, avec le même poids, ce qui implique que $\mathcal{S}$ est équivalent à $\mathcal{A}$. Dans $\mathcal{S}$, on peut effacer un certain nombre de transitions de façon à ce que, pour chaque mot accepté, il reste un et un seul chemin réussi. On obtient ainsi un automate non ambigu, et comme $\mathcal{A}$ (et donc $\mathcal{S}$) est 1-valué, cet automate réalise la même série, ce qui prouve $\overline{\mathsf{Amb}} = \mathsf{1val}$.

Voyons cette construction. L'idée est de faire en sorte que les chemins réussis qui restent après effacement des transitions soient en bijection avec ceux de $\mathcal{D}$ qui est déterministe donc *a fortiori* non ambigu.

On dit qu'il y a une *compétition* dans $\mathcal{S}$ si il existe deux états (p, P) et (q, P) qui remplissent une des deux conditions suivantes :
(a) ils sont tous deux terminaux ;
(b) de ces états partent des transitions étiquetées par la même lettre et ayant le même état d'arrivée.

On obtient l'automate $\mathcal{U}$ à partir de $\mathcal{S}$ en supprimant le minimum de transition et de flèches terminales de sorte qu'il n'y ait plus de compétition. Cette procédure implique un choix arbitraire.

Cette construction est la même que celle donné par J. Sakarovitch [51] pour les transducteurs. Dans [26], nous avons montré que si l'automate est finiment ambigu, on peut en déduire une union finie d'automates non ambigus.

On montre que si l'automate $\mathcal{A}$ est finiment ambigu, pour tout chemin C réussi de $\mathcal{A}$, il existe un choix pour les compétitions de $\mathcal{S}$ qui permet d'obtenir un automate non ambigu conservant le chemin en bijection avec C. Cela découle du fait que, sous cette hypothèse, deux transitions en compétitions de $\mathcal{S}$ n'appartiennent jamais au même chemin réussi.

En faisant tous les choix possibles pour chaque compétition, on obtient ainsi un ensemble fini d'automates non ambigu tel que tout chemin réussi de $\mathcal{A}$ apparaît dans au moins un de ces automates. On peut évidemment un peu affiner ce choix pour restreindre le nombre d'automates non ambigus ainsi construits.

3 Problèmes de déterminisation

Tout automate max-plus n'est pas déterminisable. Un cas simple est celui où l'automate vérifie la propriété de jumelage suivante :
Pour tout mot uv, s'il existe plusieurs chemins partant d'un état initial étiquetés par ce mot et tels que v étiquette des circuits sur ces chemins, tous ces circuits ont le même poids.

Dans ce cas, l'automate est déterminisable et la construction d'un automate déterministe équivalent est similaire à celle d'un transducteur séquentielle à partir d'un transducteur fonctionnel vérifiant une propriété de jumelage analogue (*cf.* [12]).

En adaptant la preuve de Ch. Choffrut [12], M. Mohri [47] a montré que pour les automates non ambigus, cette propriété de jumelage est nécessaire pour la série réalisée soit séquentielle. Il n'en est rien dans le cas général.

Le travail effectué sur les automates max-plus a pour but de trouver une caractérisation complète des automates déterminisables. Pour l'instant cette question reste largement ouverte. Une autre propriété importante de ces automates est la non ambiguïté. En effet, décider la non ambiguïté permet, à condition que la procédure de décision permette aussi la construction le cas échéant d'un automate non ambigu équivalent, de décider la déterminisabilité. Cette question est elle aussi ouverte dans le cas général.

Avec I. Klimann, J. Mairesse et Ch. Prieur, nous avons montré le résultat suivant :

Théorème 5.1 *[25, 26] Si $\mathcal{A}$ est un $\mathfrak{R}$-automate finiment ambigu, on peut décider si la série réalisée par $\mathcal{A}$ est réalisable par un automate non ambigu.*

C'est ce résultat que je vais présenter ici, ainsi que la construction de l'automate non ambigu qui en résulte.

La propriété de dominance La construction précédente permet de mettre un automate finiment ambigu sous la forme d'une union d'automates non ambigus de même support.

On se ramène donc à décider si la série réalisée par une union de n automates non ambigus est réalisable par un seul automate non ambigu. Considérons $\mathcal{P}$ l'automate obtenu par produit cartésien de ces n automates, et dont les poids sont des n-uplets. On définit successivement :

- *Les coordonnées victorieuses d'un circuit* de $\mathcal{P}$ est l'ensemble des coordonnées dont le poids est maximal sur ce circuit ;
- l'ensemble des coordonnées victorieuses d'une *composante fortement connexe* est l'intersection des coordonnées victorieuses des circuits simples de cette composante ;
- l'ensemble des coordonnées victorieuses *d'un chemin* est l'intersection des coordonnées victorieuses des composantes fortement connexes traversées par ce chemin.

Si un chemin ne traverse aucune composante fortement connexe, toutes ses coordonnées sont victorieuses.

L'automate produit $\mathcal{P}$ vérifie la propriété de dominance si tout chemin réussi a un ensemble de coordonnées victorieuses non vide.

Proposition 5.1 *Soit $\mathcal{A}_i$ un ensemble fini de $\mathfrak{R}$-automates non ambigus. L'union des automates $\mathcal{A}_i$ réalise une série non ambiguë si et seulement si l'automate produit des $\mathcal{A}_i$ vérifie la propriété de dominance.*

Soit M la différence entre le plus grand et le plus petit coefficient de l'automate et n le nombre d'états de $\mathcal{P}$. Pour tout vecteur α, on note $\breve{\alpha}$ la plus petite valeur non nulle (*i.e.* différente de $-\infty$) et $\bar{\alpha} = \alpha - \breve{\alpha}$. On construit l'automate $\mathcal{U}$ suivant :

- pour tout état initial p de $\mathcal{P}$ de poids α, $(p, \bar{\alpha})$ est un état initial de $\mathcal{U}$ de poids $\breve{\alpha}$;
- pour toute transition (p, a, α, q) de $\mathcal{P}$, pour tout état (p, β) de $\mathcal{U}$, il y a un transition de (p, β) vers l'état $(q, \bar{\gamma})$ de $\mathcal{U}$, avec
$$\gamma_i = \begin{cases} -\infty & \text{si } \beta_i + \alpha_i < \beta_j + \alpha_j - Mn \\ \beta_i + \alpha_i & \text{sinon ;} \end{cases}$$
où j est une coordonnées victorieuse de la composante fortement connexe de q ayant un poids non nul minimal dans $\beta + \alpha$.
- pour tout état terminal p de $\mathcal{P}$ de poids final α, pour tout état (p, β) de $\mathcal{U}$, (p, β) est terminal de poids $\mathsf{max}(\beta_i + \alpha_i)$.

Si $\mathcal{P}$ vérifie la propriété de dominance, on montre que cet automate est équivalent à l'union des $\mathcal{A}_i$. On montre en outre que la propriété sur $\mathcal{P}$ est nécessaire pour que la série réalisée soit non ambiguë.

Exemple. L'automate $\mathcal{A}_2$ ci dessous est finiment ambigu :

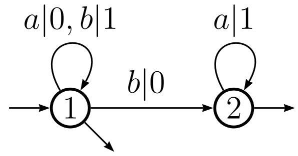


Fig. 5.1- Un automate finiment ambigu

Afin de le transformer en union finie d'automates non ambigus, on calcule le revêtement de Schützenberger :

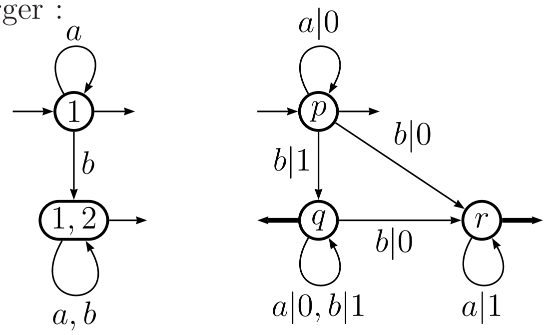


Fig. 5.2- Revêtement de Schützenberger de $\mathcal{A}_2$

À gauche figure le déterminisé de $\mathcal{A}_2$. Seules les deux flèches terminales du bas sont en compétition ; les deux automates sont donc obtenus en enlevant l'une ou l'autre :

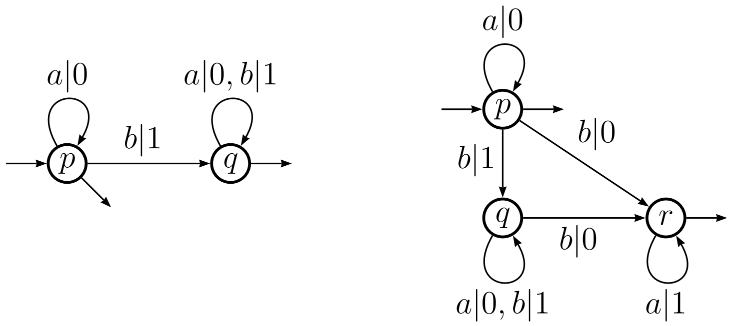


Fig. 5.3- Deux automates non ambigus $\mathcal{A}_2^{(1)}$ et $\mathcal{A}_2^{(2)}$

On calcule maintenant le produit de ces deux automates :

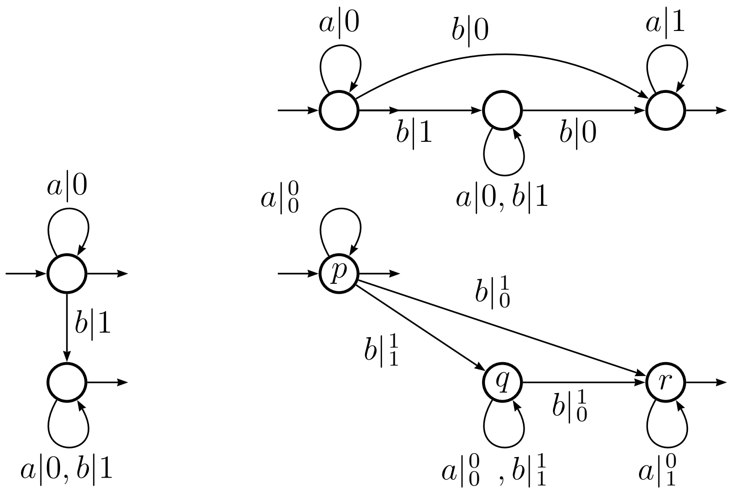


Fig. 5.4- L'automate produit de $\mathcal{A}_2^{(1)}$ et $\mathcal{A}_2^{(2)}$

La seule composante fortement connexe dans laquelle une coordonnée domine l'autre est la dernière boucle, dans laquelle la seconde composante est victorieuse. Donc, pour tout chemin réussi dans le produit, l'ensemble des composantes victorieuses est soit l'ensemble $\{1,2\}$ (pour les mots de a^*), soit l'ensemble $\{2\}$; la propriété de dominance est vérifiée.

On calcule alors un automate non ambigu selon la construction donnée plus haut :

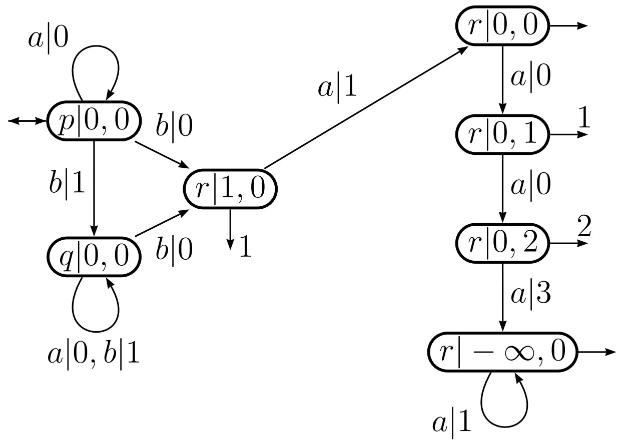


FIG. 5.5- Un automate non ambigu équivalent à $\mathcal{A}_2$

On le voit, à un moment, la différence entre les deux valeurs du vecteur stocké devient trop grande ; on met donc la plus petite à $-\infty$.

4 Non ambiguïté et max/min-plus reconnaissance

Nous l'avons vu à la section précédente, la famille des automates non ambigus est importante car la plupart des problèmes de décision y sont calculables, notamment la déterminisabilité et l'équivalence.

Nous donnons ici une caractérisation des séries réalisables par automate max-plus non ambigu :

Théorème 5.2 *[36] Une série est réalisable par un automate max-plus non ambigu si et seulement si elle est à la fois réalisable par un automate max-plus et par un automate min-plus.*

Ce résultat unifie deux familles de séries rationnelles et répond à un problème ouvert par D. Krob [29] : caractériser les séries reconnaissables à la fois par automate max-plus et par automate min-plus.

Il est facile de voir qu'une série max-plus réalisée par un automate non ambigu est aussi une série rationnelle min-plus ; il suffit de considérer l'automate max-plus comme automate min-plus, la non ambiguïté fait que l'on n'utilise pas la loi additive du semi-anneau.

Pour la réciproque, considérons une série S réalisée à la fois par un automate max-plus $\mathcal{A} = (I, \mu, T)$ et un automate min-plus $\mathcal{B} = (J, \nu, U)$. Pour un mot w donné, chaque chemin réussi de $\mathcal{A}$ étiqueté par w a un poids inférieur à $\langle S, w\rangle$ et chaque chemin réussi de $\mathcal{B}$ étiqueté par w a un poids supérieur à $\langle S, w\rangle$.

Si l'on transforme le poids de chaque transition de $\mathcal{B}$ en en prenant l'opposé, on obtient un automate max-plus $\mathcal{B}'$ tel que chaque chemin réussi de $\mathcal{B}'$ étiqueté par w a un poids inférieur à $-\langle S, w\rangle$.

Le produit cartésien des automates $\mathcal{A}$ et $\mathcal{B}'$ donne donc un automate max-plus $\mathcal{C}$ dont chaque chemin réussi étiqueté par w a un poids inférieur à 0. Comme par ailleurs il existe dans $\mathcal{A}$ (*resp.* $\mathcal{B}$) au moins un chemin étiqueté par w de poids $\langle S, w\rangle$ (*resp.* $-\langle s, w\rangle$), $\mathcal{C}$ réalise la série nulle.

On peut donc faire circuler les coefficients dans $\mathcal{C}$ de façon à n'avoir que des poids négatifs. En gardant uniquement les transitions de poids 0, on obtient un automate 1-valué, et, en reportant sur ces transitions les poids des transitions correspondantes de $\mathcal{A}$, on obtient un automate 1-valué qui réalise S.

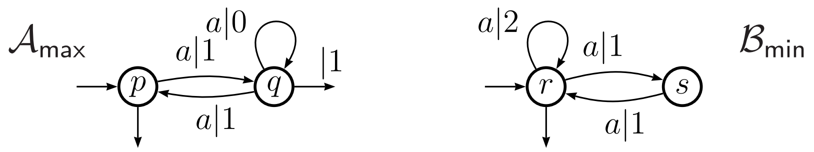
Le résultat utilisé ci-dessus est une propriété de Fatou : si une série rationnelle $\mathfrak{R}$ a des coefficients négatifs, c'est aussi une série rationnelle dans $\mathfrak{R}^-$.

En effet, soit $\mathcal{A} = (I, \mu, T)$ un automate qui réalise une telle série. Posons

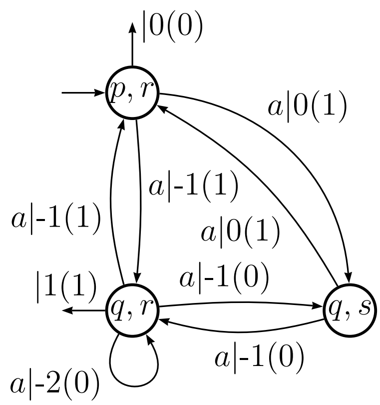
$$M = \bigoplus_{a\in A} \mu(a), \quad X = \bigoplus_{k=0}^{n} M^k \quad \text{et} \quad \alpha = IX.$$

Alors en faisant circuler α_p à travers p, on obtient un automate équivalent à poids négatifs. Cette construction, prouvée en [36], étend un résultat de D. Krob [29].

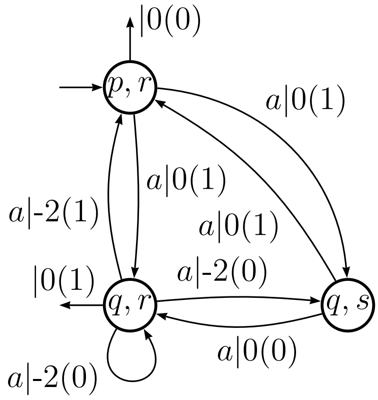
Exemple. Considérons l'automate max-plus $\mathcal{A}_{\mathsf{max}}$ et l'automate min-plus $\mathcal{B}_{\mathsf{min}}$ ci-dessous :



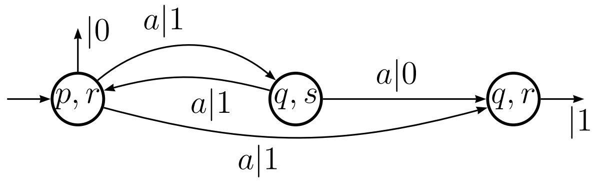
On fait le produit des deux automates en reportant la différence des poids sur chaque transition (entre parenthèses, on reporte les poids de $\mathcal{A}_{\mathsf{max}}$).



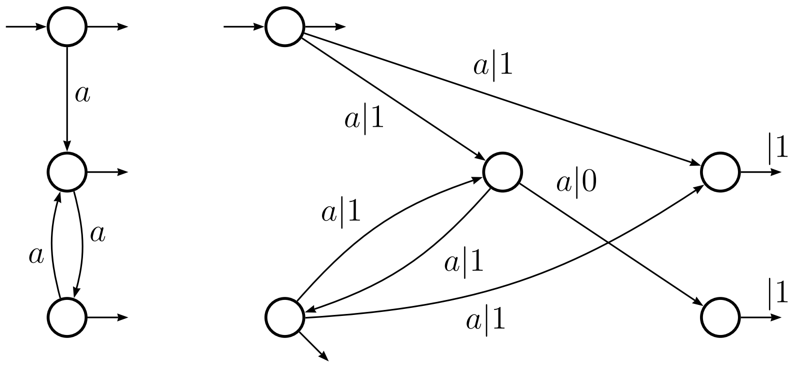
Le vecteur α indiquant les coefficients à faire circuler est égal à $[0 \;\; -1 \;\; 0]$; il faut donc faire circuler -1 à travers l'état (q, r), ce qui donne l'automate suivant :



On supprime maintenant toutes les transitions qui ont un poids strictement négatif et on reporte les poids de $\mathcal{A}_{\mathsf{max}}$:



On obtient un automate 1-valué qui réalise la même série que $\mathcal{A}_{\mathsf{max}}$ et $\mathcal{B}_min$. Bien qu'on puisse ici rendre très simplement cet automate non ambigu, appliquons la construction systématique :



À gauche figure le déterminisé de l'automate sous-jacent.

La seule compétition concerne les deux flèches finales de la dernière ligne ; il suffit d'en supprimer une pour obtenir un automate non ambigu.

Chapitre 6

Vaucanson

1 La plate-forme Vaucanson

Le projet VAUCANSON est né, sous sa forme actuelle en 2002. Le but était de produire une bibliothèque performante permettant de manipuler différents types d'automates à multiplicité. Le cahier des charges comprend différentes contraintes antagonistes. D'une part, on veut qu'un algorithme "générique" (accessibilité par exemple) ne soit écrit qu'une fois et soit applicable sur tous les automates que l'on va définir, quelque soit leurs multiplicités, voir leur implantation. Cette contrainte conduit à une approche objet de la plate-forme. Par ailleurs, on veut conserver de bonnes performances, ce qui nécessite en particulier de ne pas avoir à résoudre les typages des fonctions lors de l'exécution.

La solution choisie a donc été la généricité statique : chaque fonction générique est en fait instanciée lors de la compilation pour obtenir du code dédié au type d'automate manipulé. Le fonctionnement détaillé de l'architecture se trouve en [37, 38].

2 Quelques algorithmes

Ma contribution à ce projet, outre l'encadrement des étudiants procédant à l'implantation et des conseils concernant la théorie des automates, se situe au niveau des algorithmes que j'ai parfois implantés ou adaptés, voire inventés.

Je vais ici en présenter quelques uns, soit pour leur intérêt propre, soit parce qu'ils sont révélateurs de la différence entre théorie et pratique.

2.1 Dérivation des expressions rationnelles

Il s'agit évidemment d'expressions rationnelles avec multiplicités.

Une expression rationnelle est représentée dans VAUCANSON par un arbre dont chaque nœud interne représente un opérateur et chaque feuille une lettre.

J.-M. Champarnaud et D. Ziadi [11] ont montré que toute dérivée d'une expression rationnelle E peut toujours s'exprimer comme produit de sous-expressions de E. Si on dispose de pointeurs sur chaque nœud de l'expression, une dérivée est donc codée par une suite de pointeurs. De plus, on peut disposer des liens dans l'arbre de la dérivée de façon à calculer rapidement cette suite de pointeurs. C'est cette méthode que nous avons adaptée aux expressions avec multiplicités.

– Pour tout nœud “$*$” n, il y a un lien du fils de n à n ;
– pour tout nœud “$\cdot$” n, il y a un lien du fils gauche de n vers son fils droit.

Pour dériver une expression E par rapport à a, on recherche dans E les lettres a qui peuvent débuter un mot, on remonte de chacune de ces lettres vers la racine de E en collectant les extrémités des liens dont on rencontre le départ.

Exemple. Considérons l'expression $\mathsf{E}_5 = (2ab + (3b) \cdot (4ab)^*)^*$ déjà étudiée dans le chapitre 3. L'arbre de cette expression est le suivant :

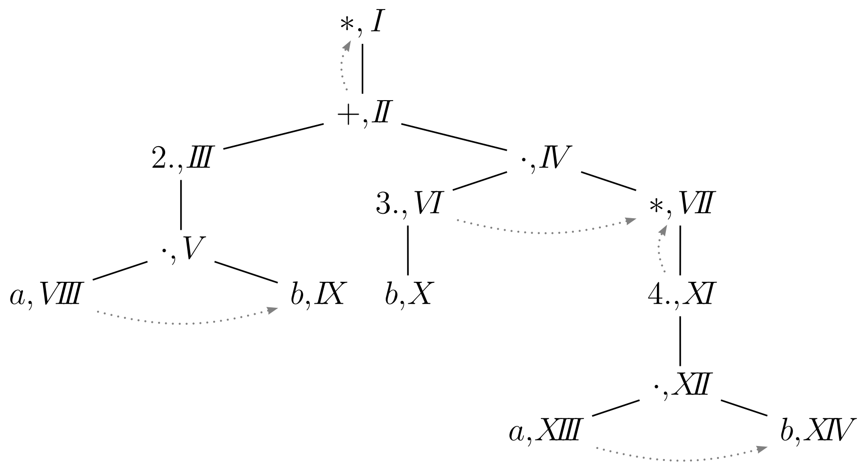


Fig. 6.1- L'arbre de l'expression E_5

L'expression E_5 est représenté par la liste $[I]$. Pour dériver cette expression par a, on recherche les a accessibles en tête à partir de I avec leur multiplicité. On trouve VIII, avec multiplicité 2. En remontant, on rencontre deux liens dont les extrémités se trouvent, dans l'ordre des rencontres en IX et I. En appliquant une procédure similaire pour toutes les dérivées, on trouve :

$$\frac{\partial}{\partial a}[I] = 2[\mathit{IX}; I] = 2\mathsf{K}_1 \qquad \frac{\partial}{\partial b}[I] = 3[\mathit{VII}; I] = 3\mathsf{K}_2$$
$$\frac{\partial}{\partial a}[\mathit{IX}; I] = 0 \qquad \frac{\partial}{\partial b}[\mathit{IX}; I] = I = \mathsf{E}_5$$
$$\frac{\partial}{\partial a}[\mathit{VII}; I] = 4[\mathit{XIV}; \mathit{VII}; I] = 4\mathsf{K}_3 \qquad \frac{\partial}{\partial b}[\mathit{VII}; I] = 0$$
$$\frac{\partial}{\partial a}[\mathit{XIV}; \mathit{VII}; I] = 0 \qquad \frac{\partial}{\partial b}[\mathit{XIV}; \mathit{VII}; I] = [\mathit{VII}; I] = \mathsf{K}_2$$

Trois fonctions sont donc requises et implantées ; elles prennent comme paramètre l'adresse d'un nœud dans l'arbre. La première retourne le poids du mot vide dans la série représentée par le sous-arbre ; la seconde donne pour chaque lettre a une liste de couples pointeurs/poids indiquant les feuilles a accessibles en tête avec leur multiplicité ; la troisième donne une liste de pointeur obtenue en remontant à partir du nœud indiqué.

La construction de l'automate suit ensuite celle donnée au chapitre 3.

2.2 Minimisation

La minimisation des automates déterministes est un problème intéressant en pratique, car quand on veut utiliser un automate, c'est-à-dire lui donner en entrée un texte *a priori* long, on doit le déterminiser et, pour ne pas manipuler des objets trop gros, le minimiser autant que possible. Ce problème est intéressant en théorie aussi : l'algorithme de Hopcroft [22] a une complexité $O(n \log n)$ pour minimiser les automates déterministes quelconques. Pour certaines classes très particulières, par exemple les automates acycliques, il existe des algorithmes linéaires (*cf.* [50]). En pratique, les algorithmes de meilleure complexité ne sont pas forcément les plus rapides, car les automates que l'on rencontre ne sont pas forcément en "position générale". La plate-forme VAUCANSON implémente pour l'instant l'algorithme de Moore [48] et celui d'Hopcroft [22].

C'est ce dernier que nous avons adapté au calcul du $\mathbb{K}$-co-quotient minimal. J'ai déjà défini ce qu'est un $\mathbb{K}$-quotient au chapitre 2. L'existence d'un $\mathbb{K}$-quotient minimal est assez simple ; il est assez clair que l'équivalence φ sur les états d'un $\mathbb{K}$ automate la plus grossière qui vérifie l'équation (2.1) page 19 induit une fusion dont le résultat est un $\mathbb{K}$-quotient minimal.

Le but est donc de calculer cette équivalence, ce qui se fait en deux temps :
- On initialise la partition $\mathcal{P}$ de sorte que deux états sont équivalents si et seulement si ils ont le même poids final. Chaque couple de $\mathcal{P} \times A$ (A est l'alphabet) est placé dans une file f.
- Tant que f est non vide, on en sort un couple (P, a). On considère l'ensemble des classes Q telles qu'il existe une transition d'un état q de Q à un état p de P. Chacune de ces classes Q est partitionnée de sorte que deux états q et q' de Q demeurent équivalents si et seulement si la somme de leurs transitions étiquetées par a arrivant en un état de P est la même. Si de nouvelles classes sont ainsi créées, elles sont placées dans la file avec chaque lettre.

Le principe de l'algorithme est le même que celui d'Hopcroft. Il consiste, de façon un peu surprenante à considérer non pas les transitions qui partent de la part qu'on traite, mais de celles qui arrivent. Le fait que l'on considère des automates non booléens a diverses conséquences : la partition de départ peut représenter plus de deux classes ; à chaque fois qu'on crée de nouvelles classes, on les place toutes dans la file car les arguments de tiers exclus fonctionnant sur les booléens ne sont plus vrais. Ce dernier point empêche en particulier de conserver la complexité $O(n \log n)$.

2.3 Transducteurs

Les transducteurs sont assez révélateurs des difficultés que l'on peut rencontrer quand on implémente des objets mathématiques. L'habitude nous amène à confondre des objets différents entre lesquels existe un isomorphisme naturel.

Ainsi, les transducteurs sont des automates à deux bandes, c'est-à-dire, si on veut donner une interprétation algébrique de la chose, des automates sur le produit de deux monoïdes libres. Cette description est parfaitement symétrique ; elle est de plus particulièrement pratique si on parle de transducteurs normalisés, c'est-à-dire avec des étiquettes dans $A \times \{1\} \cup \{1\} \times B$, de transducteur lettre-à-lettre (étiquettes dans $A \times B$), *etc.*

Mais il arrive souvent que l'on s'intéresse à des propriétés non symétriques, telles que les transducteurs "temps réel" (étiquettes dans $A \times \mathsf{Rat}B^*$) ou séquentiels (déterministes par rapport à la première entrée). Dans ce cas là, il est intéressant tant au niveau théorique que pratique de voir les transducteurs comme des $\mathsf{Rat}B^*$-automates, $\mathsf{Rat}B^*$ étant après tout un semi-anneau comme un autre. Chaque transition est alors étiquetée par une lettre (ou un mot) de A et a un poids dans $\mathsf{Rat}B^*$. Il n'y a alors pas besoin, par exemple, d'écrire un programme particulier pour rendre les transducteurs “temps réel”, puisque c'est la même opération que celle qui consiste à faire en sorte qu'un $\mathbb{K}$-automate quelconque ait des transitions étiquetées par des lettres. De même, sous-réserve que les bonnes propriétés soient vérifiées, la séquentialisation s'écrira comme la déterminisation des automates max-plus.

Vaucanson possède les deux implémentations ainsi que les algorithmes de passage de l'une à l'autre.

Un transducteur réalise avant tout une relation (rationnelle) entre des mots sur un alphabet A et des mots sur un alphabet B. Les opérations fondamentales que l'on peut exiger, quelle que soit la forme du transducteur, sont celles qui concernent les relations elles-même, c'est-à-dire la composition et l'évaluation. Nous avons écrit ces algorithmes pour les deux implantations. Il est amusant de voir que ces deux algorithmes sont liés mais que la relation de dépendance n'est pas dans le même sens selon la façon de considérer les transducteurs.

Si l'on considère les transducteurs comme automates dont les poids sont des ensembles de mots, le premier algorithme est l'évaluation. Tout d'abord, on met le transducteur sous la forme “temps-réel” ; on calcule le produit avec l'automate représentant le langage L dont on veut l'image ; on obtient ainsi un transducteur qui réalise la même relation, mais dont le domaine est limité au langage L. Il suffit ensuite de construire un automate booléen en remplaçant sur chaque transition l'étiquette par la sortie.

Exemple. Considérons le transducteur $\mathcal{T}_2$ suivant :

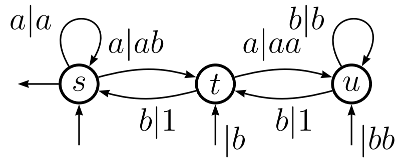


Fig. 6.2- Le transducteur $\mathcal{T}_2$

Pour évaluer l'image du langage $abb + baa$ par ce transducteur, on effectue un produit.

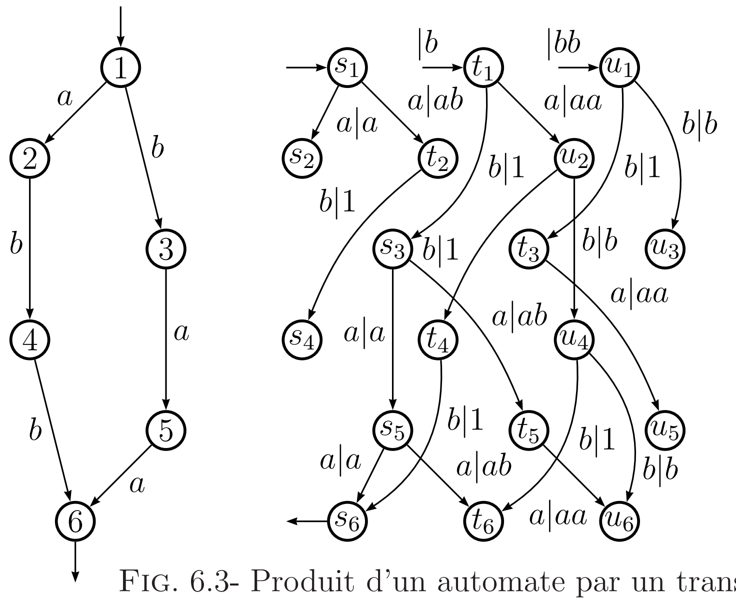


Fig. 6.3- Produit d'un automate par un transducteur

Après émondage et projection, il reste :

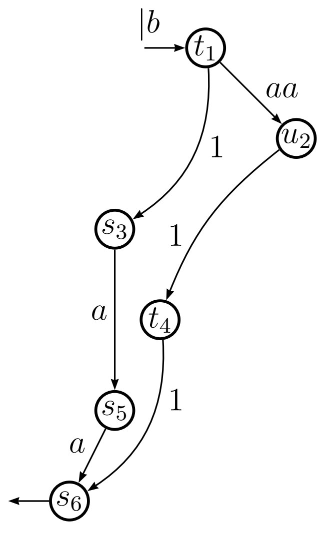


Fig. 6.4- L'image de $abb + baa$ par $\mathcal{T}_2$

Ces deux mots ont donc la même image : baa.

La composition des transducteurs "temps réels" est en fait une suite d'évaluations. D'un point de vue purement algébrique, si l'on considère les représentations linéaires, la représentation de la composition est le produit tensoriel des représentations de chacune des relations de départ. En clair, le transducteur réalisant la composition des transducteurs

$\mathcal{S} = (I, \mu, T)$ et $\mathcal{T} = (J, \nu, U)$, est $\mathcal{T} \circ \mathcal{S} = (K, \xi, V)$, où, pour tout a, $\xi(a)$ est obtenu en remplaçant chaque $\mu(a)_{i,j}$ par $\nu(\mu(a)_{i,j})$.

Exemple. Considérons le transducteur $\mathcal{S}_2$:

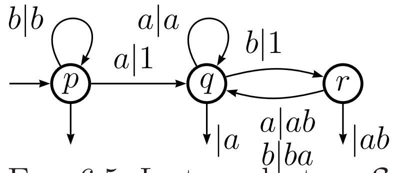
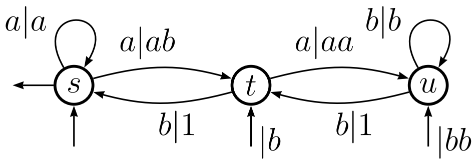


Fig. 6.5- Le transducteur $\mathcal{S}_2$

Calculons la composition de ce transducteur par $\mathcal{T}_2$ représenté figure 6.2.



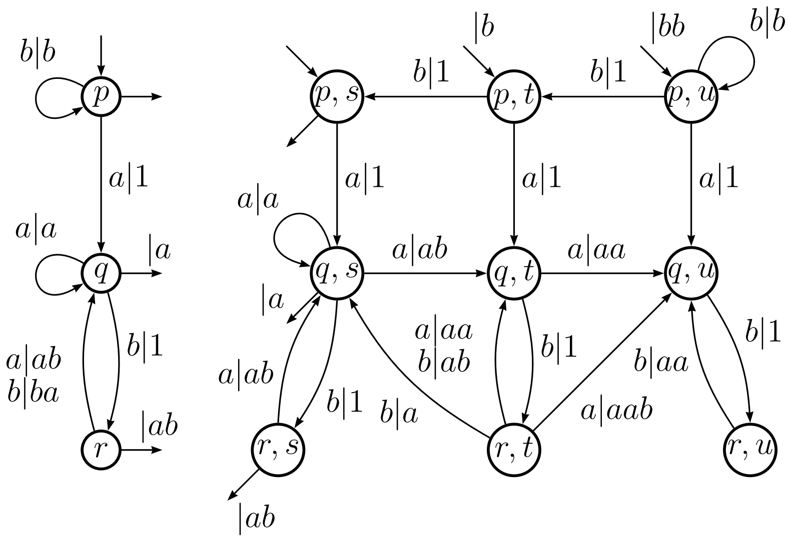


Fig. 6.6- La composition des transducteurs $\mathcal{S}_2$ et $\mathcal{T}_2$

Dans le cas des transducteurs vus comme automates sur le monoïde libre, la mise sous forme "temps réel" est moins naturelle, donc l'évaluation s'en trouve compromise. Toutefois, si l'on met les transducteurs sous forme normale, on peut calculer assez simplement un transducteur réalisant la composition. En réalité, on manipule des transducteurs sous-normalisés, c'est-à-dire dont les étiquettes sont dans $A \times \{1\}$, $\{1\} \times B$, mais aussi $A \times B$, ce qui permet de gagner un peu en concision. Examinons cet algorithme sur un exemple.

Exemple. Pour simplifier, dans cet exemples, les alphabets d'entrée et de sortie des transducteurs sont les mêmes. Considérons les deux transducteurs $\mathcal{S}_7$ et $\mathcal{T}_7$ ci-dessous :

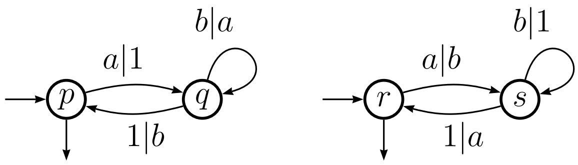


Fig. 6.7- Les transducteurs $\mathcal{S}_7$ et $\mathcal{T}_7$

Calculons la composition de ces deux transducteurs :

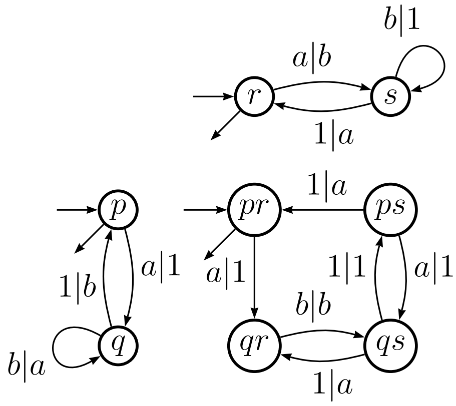


Fig. 6.8- Composition des transducteurs $\mathcal{S}_7$ et $\mathcal{T}_7$

Les transitions du composés sont construites suivant deux règles :

- Pour toute transition de $\mathcal{S}_7$ dont la sortie est le mot vide (ici $(p, a|1, q)$), on reporte une transition avec la même étiquette dans chaque "colonne" du composé. De même, pour toute transition de $\mathcal{T}_7$ dont l'entrée est le mot vide (ici $(s, 1|a, r)$), on reporte une transition avec la même étiquette dans chaque "ligne" du composé.
- Pour toute transition $(p, u|a, q)$ de $\mathcal{S}_7$ dont la sortie est une lettre, pour toute transition $(r, aw|v, s)$ de $\mathcal{T}_7$ dont l'entrée est la même lettre, on construit la transition $((p, r), u|v, (q, s))$.

Selon la nature de u et v, cette seconde opération peut donner des transitions étiquetées par $1|1$; si l'on veut retrouver un transducteur sous-normalisé, il faudra ensuite supprimer celles-ci par un algorithme classique de suppression des ε-transitions. On obtient finalement le transducteur suivant après émondage :

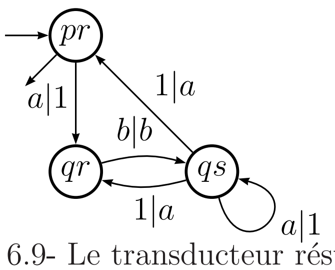


Fig. 6.9- Le transducteur résultat

Pour effectuer l'évaluation, il suffit de calculer l'automate qui reconnaît le langage dont on veut l'image, puis de remplacer chaque étiquette a par $a|a$ afin d'obtenir un transducteur réalisant l'intersection par ce langage. La composition de ce transducteur avec celui réalisant la fonction donne un transducteur à partir duquel, en projetant selon la sortie, on obtient l'automate reconnaissant l'image voulue.

Le défaut de la procédure de composition présentée ci-dessus est qu'elle ne conserve pas les multiplicités. En effet, on pourrait très bien, et VAUCANSON le permet, mettre des poids sur les transitions de ces transducteurs. Examinons par exemple le mot *abab*. Dans le transducteur $\mathcal{S}_7$, il n'y a qu'un chemin admettant ce mot pour entrée : (p, q, q, p, p, q, q, p) ; sa sortie est *abab*. De même, dans $\mathcal{T}_7$, il y a un seul chemin : (r, s, s, r, r, s, s, r) de sortie *baba*.

Par contre, dans le transducteur composé, il y a deux chemins étiquetés par *abab* : $(pr, qr, qs, qs, qr, qs, pr)$ et $(pr, qr, qs, pr, qr, qs, pr)$. Ces chemins ont (bien sûr) la même sortie *baba*. Que se passe-t-il ? En fait, dans le premier, on lit d'abord $(a, 1)$ puis $(1, a)$, alors que dans le second c'est le contraire. En effet, lorsque le premier transducteur sort le mot vide et que le second lit le mot vide en entrée, il n'y a pas de synchronisation et l'ordre des deux transitions est arbitraire. Pour palier ce défaut, on donne une priorité à l'un des deux transducteurs. On décide que si une telle situation se présente, on lit d'abord toutes les transitions du second transducteur, puis toutes celles du premier.

Pour celà, on calcule un revêtement $\mathcal{S}'$ du transducteur $\mathcal{S}$, de sorte à avoir deux types d'états : ceux dans lesquels n'entrent que des transitions dont la sortie est le mot vide, et d'autres qui peuvent être initiaux et dont toutes les transitions partantes ont une lettre en sortie. De même, on calcule un co-revêtement $\mathcal{T}'$ du transducteur $\mathcal{T}$, de sorte à avoir deux types d'états : ceux desquels ne sortent que des transitions dont l'entrée est le mot vide, et d'autres qui peuvent être terminaux et dont toutes les transitions qui arrivent ont une lettre en entrée.

EXEMPLE. Appliquons ces revêtements aux transducteurs $\mathcal{S}_7$ et $\mathcal{T}_7$.

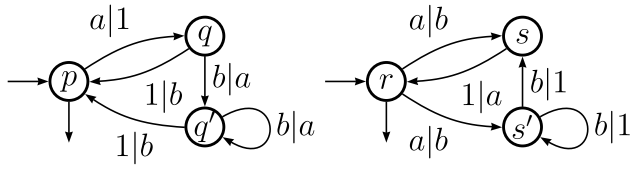


FIG. 6.10- Les transducteurs $\mathcal{S}'_7$ et $\mathcal{T}'_7$

On calcule maintenant le transducteur composé :

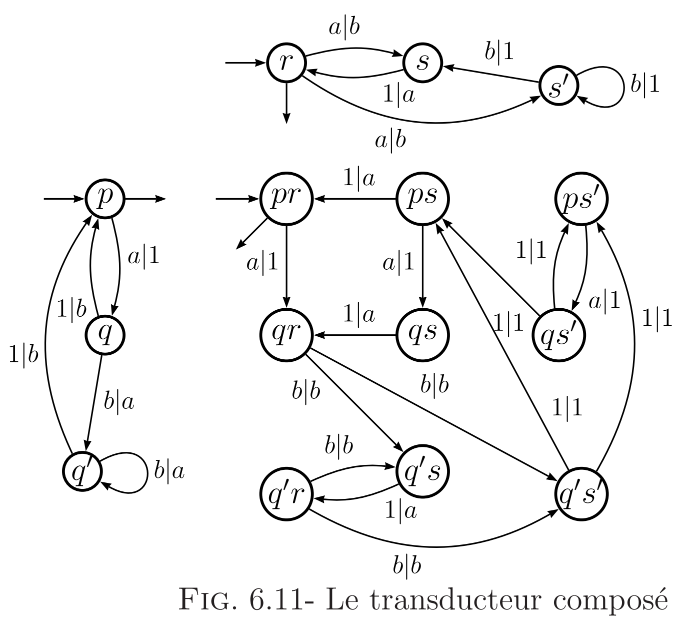


FIG. 6.11- Le transducteur composé

Dans $\mathcal{S}'_7$, quand on est en q, on vient forcément de sortir le mot vide ; de même, dans $\mathcal{T}'_7$, quand on est en s, on va forcément lire le mot vide en entrée. Il faut donc supprimer l'état qs dans le transducteur si on veut évider cette succession d'étiquettes. On obtient alors un transducteur qui respecte les chemins des deux transducteurs de départ.

Bibliographie

[1] Valentin M. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoret. Comput. Sci.*, 155(2) :291–319, 1996.

[2] André Arnold, Anne Dicky et Maurice Nivat. A note about minimal non-deterministic automata. *Bulletin of the EATCS*, 47 :166–169, 1992.

[3] Marie-Pierre Béal. *Codage symbolique*. Études et recherches en informatique. Masson, Paris, 1993.

[4] Marie-Pierre Béal, Sylvain Lombardy et Jacques Sakarovitch. On the equivalence of $\mathbb{Z}$-automata. In *ICALP 2005*, volume 3580 de *Lect. Notes Comp. Sci.*, pages 397–409. Springer, 2005.

[5] Jean Berstel. *Transductions and context-free languages*, volume 38 de *Leitfäden der Angewandten Mathematik und Mechanik*. B. G. Teubner, Stuttgart, 1979.

[6] Jean Berstel et Christophe Reutenauer. *Les séries rationnelles et leurs langages*. Études et recherches en informatique. Masson, Paris, 1984. Traduction anglaise : Rational series and their languages, Springer, 88.

[7] Jean Berstel et Christophe Reutenauer. *Rational series and their languages*, volume 12 de *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1988.

[8] Janusz A. Brzozowski. Derivatives of regular expressions. *J. Assoc. Comput. Mach.*, 11 :481–494, 1964.

[9] Pascal Caron et Marianne Flouret. Glushkov construction for series : The non commutative case. *Int. J. Comput. Math.*, 80(4) :457–472, 2003.

[10] Pascal Caron et Djelloul Ziadi. Characterization of glushkov automata. *Theoret. Comput. Sci.*, 233(1-2) :75–90, 2000.

[11] Jean-Marc Champarnaud et Djelloul Ziadi. Canonical derivatives, partial derivatives and finite automaton constructions. *Theoret. Comput. Sci.*, 289(1) :137–163, 2002.

[12] Christian Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theoret. Comput. Sci.*, 5(3) :325–337, 1977/78.

[13] Thomas Claveirole, Sylvain Lombardy, Sarah O'Connor, Louis-Noël Pouchet et Jacques Sakarovitch. Inside VAUCANSON. In *CIAA 2005*, Lect. Notes Comp. Sci. Springer, 2005. à paraître.

[14] John H. Conway. *Regular algebra and finite machines*. Mathematics series. Chapmann and Hall, London, 1971.

[15] L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10 :385–397, 1963.

[16] Samuel Eilenberg. *Automata, languages, and machines. Vol. A*, volume 58 de *Pure and Applied Mathematics*. Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1974.

[17] Samuel Eilenberg. *Automata, languages, and machines. Vol. B*, volume 59 de *Pure and Applied Mathematics*. Academic Press [Harcourt Brace Jovanovich Publishers], New York, 1976. Avec deux chapitres ("Depth decomposition theorem" et "Complexity of semigroups and morphisms") de Bret Tilson.

[18] Robert W. Floyd. Algorithm 97 : Shortest path. *Commun. ACM*, 5(6) :345, 1962.

[19] Christian Hagenah et Anca Muscholl. Computing epsilon-free nfa from regular expressions in o(n log^2(n)) time. *ITA*, 34(4) :257–278, 2000.

[20] Kosaburo Hashiguchi. Limitedness theorem on finite automata with distance functions. *J. Comput. System Sci.*, 24(2) :233–244, 1982.

[21] Kosaburo Hashiguchi. Algorithms for determining relative star height and star height. *Inform. and Comput.*, 78(2) :124–169, 1988.

[22] John E. Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In *Theory of machines and computations (Proc. Internat. Sympos., Technion, Haifa, 1971)*, pages 189–196. Academic Press, New York, 1971.

[23] Juraj Hromkovic, Sebastian Seibert et Thomas Wilke. Translating regular expressions into small epsilon-free nondeterministic finite automata. In *STACS*, Rüdiger Reischuk et Michel Morvan éditeurs, volume 1200 de *Lecture Notes in Computer Science*, pages 55–66. Springer, 1997.

[24] Daniel Kirsten. Distance desert automaton and the star height problem. *Theoret. Inform. Appl.*, 39(2) :455–509, 2005.

[25] Ines Klimann, Sylvain Lombardy, Jean Mairesse et Christophe Prieur. Deciding the sequentiality of a finitely ambiguous max-plus automaton. In *DLT 2003*, Zoltán Ésik et Zoltán Fülöp éditeurs, volume 2710 de *Lect. Notes Comp. Sci.*, pages 373–385. Springer, 2003.

[26] Ines Klimann, Sylvain Lombardy, Jean Mairesse et Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoret. Comput. Sci.*, 327(3) :349–373, 2004.

[27] Daniel Krob. Complete systems of B-rational identities. *Theoret. Comput. Sci.*, 89(2) :207–343, 1991.

[28] Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Internat. J. Algebra Comput.*, 4(3) :405–425, 1994.

[29] Daniel Krob. Some consequences of a Fatou property of the tropical semiring. *J. Pure Appl. Algebra*, 93(3) :231–249, 1994.

[30] Werner Kuich et Arto Salomaa. *Semirings, automata, languages*, volume 5 de *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1986.

[31] Hing Leung. Limitedness theorem on finite automata with distance functions : an algebraic proof. *Theoret. Comput. Sci.*, 81(1, (Part A)) :137–145, 1991.

[32] Douglas Lind et Brian Marcus. *An introduction to symbolic dynamics and coding*. Cambridge University Press, Cambridge, 1995.

[33] Sylvain Lombardy. *Approche structurelle de quelques problèmes de la thèorie des automates.* Thèse de doctorat, École nationale supérieure des télécommunications, Paris, 2001.

[34] Sylvain Lombardy. Sequentialization and unambiguity of (max,+) rational series over one letter. In *Workshop on max-plus algebras and Their Applications to Discrete-event Systems, Theoretical Computer Science, and Optimization*, Stéphane Gaubert et Jean-Jacques Loiseau éditeurs, Prague, 2001. IFAC, Elsevier Sciences.

[35] Sylvain Lombardy. On the construction of reversible automata for reversible languages. In *ICALP 2002*, Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz et Ricardo Conejo éditeurs, volume 2380 de *Lect. Notes Comp. Sci.*, pages 170–182. Springer, 2002.

[36] Sylvain Lombardy et Jean Mairesse. Series which are both max-plus and min-plus rational are unambiguous. *Theoret. Inform. Appl.*, 2005.

[37] Sylvain Lombardy, Raphaël Poss, Yann Régis-Gianas et Jacques Sakarovitch. Introducing VAUCANSON. In *CIAA 2003*, Oscar H. Ibarra et Zhe Dang éditeurs, volume 2759 de *Lect. Notes Comp. Sci.*, pages 96–107. Springer, 2003.

[38] Sylvain Lombardy, Yann Régis-Gianas et Jacques Sakarovitch. Introducing VAUCANSON. *Theoret. Comput. Sci.*, 328(1-2) :77–96, 2004.

[39] Sylvain Lombardy et Jacques Sakarovitch. Derivation of rational expressions with multiplicity. In *MFCS 2002*, Krzysztof Diks et Wojciech Rytter éditeurs, volume 2420 de *Lect. Notes Comp. Sci.*, pages 471–482. Springer, 2002.

[40] Sylvain Lombardy et Jacques Sakarovitch. Star height of reversible languages and universal automata. In *LATIN 2002*, Sergio Rajsbaum éditeur, volume 2286 de *Lect. Notes Comp. Sci.*, pages 76–90. Springer, 2002.

[41] Sylvain Lombardy et Jacques Sakarovitch. On the star height of rational languages : a new presentation for two old results. In *Words, languages & combinatorics, III (Kyoto, 2000)*, pages 266–285. World Sci. Publishing, 2003.

[42] Sylvain Lombardy et Jacques Sakarovitch. How expressions can code for automata. In *LATIN 2004*, Martin Farach-Colton éditeur, volume 2976 de *Lect. Notes Comp. Sci.*, pages 242–251. Springer, 2004.

[43] Sylvain Lombardy et Jacques Sakarovitch. Derivatives of rational expressions with multiplicity. *Theoret. Comput. Sci.*, 332 :141–177, 2005.

[44] Sylvain Lombardy et Jacques Sakarovitch. How expressions can code for automata. *Theoret. Inform. Appl.*, 39 :217–237, 2005.

[45] Robert McNaughton. The loop complexity of pure-group events. *Information and Control*, 11 :167–176, 1967.

[46] Robert McNaughton et Hisao Yamada. Regular expressions and state graphs for automata. *Trans. on electronic computers*, 9 :39–47, 1960.

[47] Mehryar Mohri. Finite-state transducers in language and speech processing. *Comput. Linguist.*, 23(2) :269–311, 1997.

[48] Edward F. Moore. Gedanken-experiments on sequential machines. In *Automata studies*, volume 34 de *Annals of mathematics studies*, pages 129–153. Princeton University Press, Princeton, N. J., 1956.

[49] Jean-Eric Pin. On reversible automata. In *LATIN*, Imre Simon éditeur, volume 583 de *Lect. Notes Comp. Sci.*, pages 401–416. Springer, 1992.

[50] Dominique Revuz. Minimisation of acyclic deterministic automata in linear time. *Theoret. Comput. Sci.*, 92(1) :181–189, 1992.

[51] Jacques Sakarovitch. A construction on finite automata that has remained hidden. *Theoret. Comput. Sci.*, 204(1-2) :205–231, 1998.

[52] Jacques Sakarovitch. *Éléments de théorie des automates.* Les classiques de l'informatique. Vuibert, Paris, 2003. Traduction anglaise à paraître, Cambridge University Press.

[53] Vaucanson. `www.lrde.epita.fr/cgi-bin/twiki/view/vaucanson`.

[54] Andreas Weber. Finite-valued distance automata. *Theoret. Comput. Sci.*, 134(1) :225–251, 1994.